Universidad Carlos III de Madrid

Escuela Politécnica Superior

Ingeniería Técnica en Informática de Gestión

Proyecto Final de Carrera

Servidor de comunicaciones para el turismo

Autor: Víctor Manuel Rincón de Luis

Tutor: Prof. D. Javier Ortiz Laguna

Fecha: 28 de Julio de 2011

Título: Aplicación

Asunto: Memoria del Proyecto de Fin de Carrera de Ingeniería Técnica en Informática de Gestión

Autor: D. Víctor Manuel Rincón de Luis

Tutor: Prof. D. Javier Ortiz Laguna

Universidad Carlos III de Madrid

Campus de Leganés

AGRADECIMIENTOS

- A Johanna por apoyarme, animarme y soportarme durante el desarrollo de este proyecto.
- A mis padres y mi hermana, por apoyarme a lo largo de toda mi vida en mis estudios y en el resto de proyectos de mi vida.
- A mi tutor Javi, por su tiempo, dedicación y comprensión a lo largo de este proyecto.
- A mis compañeros con los que compartí horas de teoría y práctica y que me ayudaron a disfrutar de una de las etapas más intensas de mi vida.
- A mis profesores por el tiempo dedicado, el cuál posiblemente valoro más ahora que entonces.
- Al movimiento Erasmus, porque me cambió la vida para siempre.
- A Finlandia, especialmente a Åland, porque me hizo ser lo que soy y siempre la llevaré muy dentro de mí.

ÍNDICE

1. INTRODUCCIÓN

1.1 Marco del proyecto fin de carrera.

El presente proyecto fin de carrera *Servidor de comunicaciones para el turismo* se engloba dentro de los proyectos de comunicación de subsistemas dentro de la Informática.

En los orígenes de la Informática como ciencia, las aplicaciones informáticas estaban orientadas a una máquina concreta con el fin de resolver un problema concreto. Con el paso del tiempo, se fueron desarrollando aplicaciones que comunicaban distintos ordenadores en distintas plataformas a través de redes de comunicación. Hoy en día gran número de aplicaciones se desarrollan para integrar servicios localizados en diversos lugares. Este tipo de integraciones permite ofrecer al usuario final productos más personalizados a fin de satisfacer sus necesidades. El presente proyecto pertenece a este último grupo y se enmarca dentro del proyecto SAMAP [1] (Sistema Adaptativo Multi-Agente de Planificación dependiente del contexto). El proyecto SAMAP tiene como objetivo construir una aplicación informática para facilitar la visita de diferentes usuarios a diferentes ciudades. Esta aplicación integrará diversos módulos que dinámicamente capturarán información existente en Internet para ayudar al usuario a la hora de elaborar sus planes de visita a destinos turísticos.

El sector turístico ha sido históricamente uno de los principales motores de la economía nacional. En el siguiente gráfico vemos que el peso del PIB (Producto Interior Bruto) turístico con respecto al PIB total se ha situado en los últimos años levemente por encima del 10%.

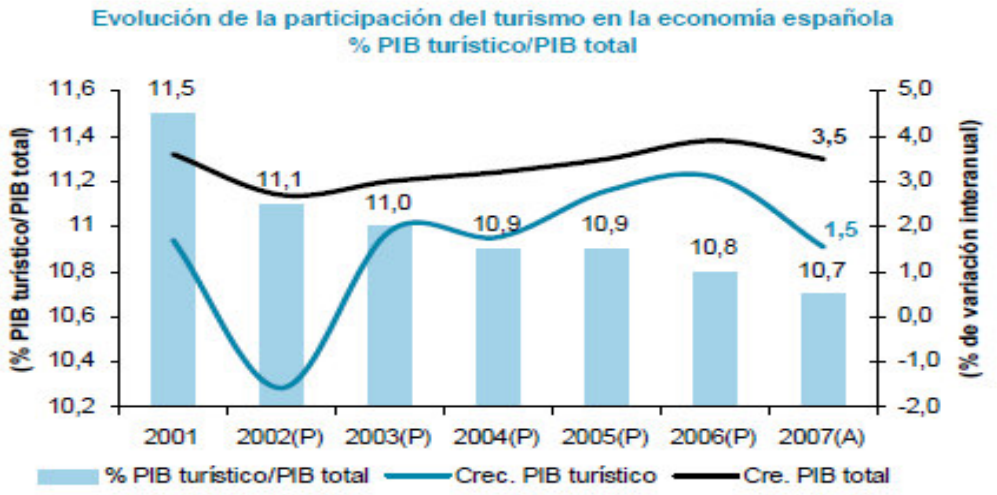

Figura 1 - Evolución de la participación del turismo en la economía española

Con mayor frecuencia se observa la presencia de las Tecnologías de la Información dentro del sector turístico, habiéndose producido un gran cambio en el modelo de negocio del sector. A la hora de planificar las vacaciones se ha pasado de consultar catálogos en papel y agencias de viajes a consultar en la red destinos, recursos y atracciones de índole turística. El billete de papel ha dado paso al billete electrónico e incluso al billete en un mensaje de texto al dispositivo móvil. No es por tanto de extrañar que el turista, en lugar de contactar o personarse físicamente en la correspondiente oficina de información turística, prefiera realizar una consulta a través de su computador personal o de su dispositivo móvil.

En la figura número 2 vemos como en los últimos años el sector turístico ha experimentado una leve crisis, posiblemente causada por la situación económica que se vive a nivel global. Debido a esta coyuntura económica, es necesario ir más allá de lo que históricamente se consideraba como “normal” en la calidad del servicio. Este grado extra de calidad, se entiende como la adaptabilidad a las necesidades concretas del usuario, ofertando flexibilidad y personificación, de tal modo que el cliente logre la sensación de estar siendo tratado de modo especial por el proveedor de servicios turísticos.

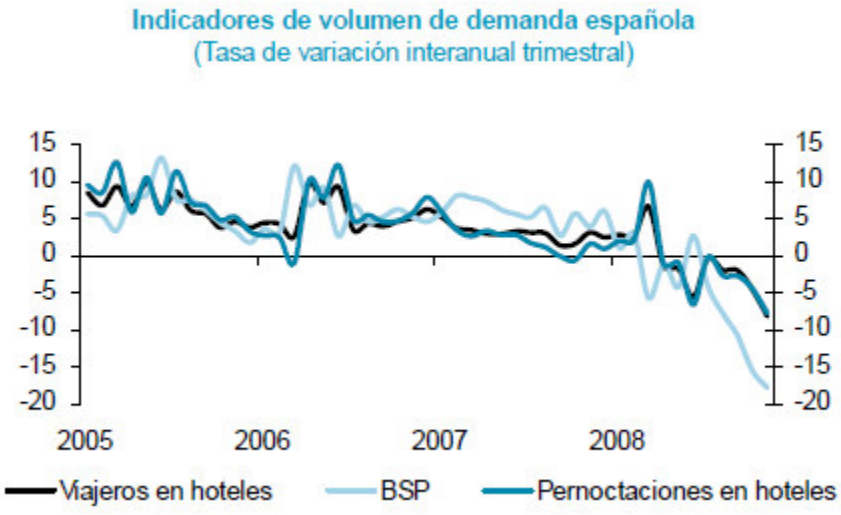

Figura 2 - Indicadores de volumen de demanda nacional

1.2 Objetivo del proyecto de fin de carrera.

El presente proyecto pretende crear una aplicación web que comunique diversos servicios orientados al sector turístico y los presente al usuario final siguiendo dos principios; movilidad, puesto que el usuario realizará consultas desde un dispositivo móvil, y precisión en la información, puesto que la aplicación deberá ofrecer al usuario la información que mejor se adapte a sus necesidades.

Al mismo tiempo se deberá construir un módulo del proyecto SAMAP que permita gestionar las comunicaciones entre los diversos módulos del mismo, de tal modo que la información facilitada a la aplicación web se ajuste siempre a las necesidades del usuario.

1.3 Estructura del presente documento.

El presente documento, describe y detalla la realización del proyecto final de carrera, estructurándolo en una serie de capítulos o apartados:

- **Capitulo 1: Introducción**

Descripción general del proyecto y de los objetivos que se persiguen con el mismo, así como la estructura que presenta la memoria.

- **Capitulo 2: Estado de la cuestión**

Descripción general de las tecnologías existentes en el mercado relacionadas con el presente proyecto fin de carrera y de aquellas soluciones que se aproximan a la desarrollada en el presente proyecto

- **Capítulo 3: Gestión del proyecto**

Definición de las fases del proyecto y cálculo de los costes del mismo, tanto en recursos humanos como de otro tipo.

- **Capítulo 4: Análisis del proyecto**

Análisis de requisitos, estudio del resto de módulos que componen el sistema y de sus interfaces de comunicación.

- **Capítulo 5: Diseño e implementación del sistema**

Diseño a bajo nivel del sistema e implementación del mismo. Desglose de cada uno de los componentes que conforman la solución implementada. Presentación del manual de usuario.

- **Capítulo 6: Resultados**

Valoración de los resultados obtenidos y confrontación con los objetivos del presente proyecto.

- **Capítulo 7: Conclusiones**

Presentación de las conclusiones generales del proyecto de fin de carrera.

- **Capítulo 8: Futuras líneas de trabajo**

Descripción de los posibles caminos a tomar en el futuro para dotar de funcionalidad añadida a la solución implementada.

- **Capítulo 9: Referencias**

Índice de referencias empleadas para la elaboración del presente proyecto fin de carrera.

- **Capítulo 10: Ilustraciones**

Índice de ilustraciones empleadas en el presente documento.

- **Capítulo 11: Tablas**

Índice de tablas empleadas en el presente documento.

2. ESTADO DE LA CUESTIÓN

En el mercado podemos encontrar gran variedad de sitios web que ofrecen servicios turísticos al turista, orientados sobre todo a realizar reservas hoteleras, o de viajes, permitiendo llevar a cabo reservas flexibles e incluso con escalas. La existencia de estos sitios permite planificar de antemano ciertos parámetros del itinerario turístico, tales como el transporte o el alojamiento, pero en lo que se refiere a la planificación de las atracciones o monumentos a visitar, la oferta, tal vez no sea tan amplia.

Otro punto a tener en consideración es que aunque la variedad de recursos web orientados al turismo es amplia, el número de sitios web especialmente diseñados para dispositivos móviles es reducido, a pesar de la existencia de aplicaciones que a partir de una página web convencional pueden crear una web especialmente diseñada para dispositivos móviles. Ejemplos de cómo poder adaptar contenidos web a dispositivos móviles pueden encontrarse en las páginas web de Mofuse [2] Infogin [3] o Momac [4].

No obstante, se pueden encontrar productos en el mercado que se acercan al modelo deseado por este proyecto. A continuación enumeraremos algunos casos de soluciones informáticas existentes en el sector turístico que están orientadas a la tecnología móvil en mayor o menor medida.

2.1 mensajes de tipo wap-push: El caso Baturamobile.

El concepto de mensajería WAP-push dentro del mundo de los dispositivos móviles, consiste en el envío de mensajes con código binario a dispositivos móviles que son capaces de interpretar este código y que soportan el estándar WAP. Normalmente dentro del mensaje se puede enviar texto, direcciones url, imágenes u otro tipo de archivos que el terminal sabe interpretar de un modo específico.

La empresa Baturamobile [5] ofrece una plataforma para distribución de información a través de la telefonía móvil. Uno de sus clientes es la ciudad de Peñíscola. El procedimiento por el cual los turistas pueden acceder a la información es mandando un mensaje de texto a un determinado número corto con un cierto código (en concreto TP para la ciudad de Peñíscola). Tras realizar esta solicitud el usuario recibirá un mensaje de tipo WAP-Push. Dentro del mensaje binario, estará incluida una dirección url a través de la cual el usuario podrá descargar una aplicación implementada

con JAVA ME (una versión reducida de JAVA SE especialmente diseñada para dispositivos móviles.)

Ciertamente este tipo de solución proporciona al usuario final información sobre el destino turístico deseado, sus recursos y atracciones, pero a un nivel general, sin estar personalizada para las necesidades o deseos concretos de la persona que planea las vacaciones.

2.2 Peticiones http: El caso Traveldodo.

En la página web de Traveldodo [6] se puede encontrar un ejemplo parecido de sistemas de distribución de información turística para dispositivos móviles.

En este caso el usuario no necesita enviar código alguno, sino que simplemente se puede descargar la guía de la ciudad de destino en formato de aplicación Java a través de un navegador web mediante una simple petición http, del tipo

http://traveldodo.com/mobile/sweden/stockholm

Una vez enviada esta petición y tras confirmar la descarga de la aplicación JAVA ME, el usuario puede acceder a comentarios *off-line* que otros usuarios de Traveldodo han dejado en la página web.

En este caso, tal vez se haya dado un paso más en el objetivo de poder planificar el viaje o las vacaciones, puesto que además de la información objetiva sobre los destinos turísticos, el usuario final dispone de los comentarios subjetivos de otros usuarios de Traveldodo, lo cual puede que le ayuden a la hora de decidir el destino a visitar. No obstante esta solución, sigue sin ofrecer excesiva flexibilidad al usuario a la hora de planificar el itinerario concreto a realizar en el destino turístico deseado.

2.3 Planificadores para el transporte: El caso *Helsinki Region Transport*.

Ciertamente existen multitud de planificadores dentro del mundo de los transportes. Uno de los que destacan por su complejidad y flexibilidad es el del servicio de transportes del municipio de Helsinki en Finlandia [7]. Este servicio permite calcular la ruta más adecuada para desplazarse empleando todos los transportes públicos de la zona metropolitana de Helsinki. En este modelo la complejidad empieza a crecer puesto que se combinan trayectos en autobús, tranvía, metro o a pie. Así mismo el sistema es capaz de informar de cortes en el servicio ocasionados por alguna situación anómala

(por ejemplo una huelga, un fallo técnico o un recorrido alternativo por obras en la línea).

Algunos lugares de interés turístico pueden encontrarse dentro de este servicio, pero desafortunadamente no permite definir rutas que pasen por distintos lugares de interés, sino simplemente crear rutas con un origen y un fin.

Este modelo tendería más hacia nuestro objetivo, ya que a la hora de analizar el recorrido óptimo entre dos puntos, la aplicación permite definir ciertos criterios de búsqueda tales como:

- medio de locomoción a emplear,
- margen de seguridad en minutos para cada cambio de transporte,
- velocidad al caminar en caso de realizar tramos a pie,
- número de recorridos alternativos a mostrar.

No obstante el usuario sigue sin poder realizar búsquedas del tipo, “bar de rock” “comida tradicional finlandesa” o “galería de arte”.

El presente proyecto pretende fusionar las aproximaciones presentadas en este apartado, de tal modo que el usuario disponga de un servicio orientado a su dispositivo móvil, permitiéndole no sólo descargar una aplicación con información estática sobre el destino deseado, sino que también le permita comprobar en tiempo real la existencia de acontecimientos de última hora que fuesen de su interés o cambios que pudiesen alterar la planificación realizada.

2.4 Características específicas de los dispositivos móviles.

A la hora de navegar por un sitio web a través de un dispositivo móvil existen ciertos aspectos que difieren con la navegación que un usuario puede realizar a través de un ordenador convencional bien sea de escritorio o portátil. Simplemente desde el punto de vista psicológico, la actitud del usuario a la hora de navegar es diferente dado que no es lo mismo estar sentado en tu escritorio con tu pantalla de tamaño aceptable, que estar caminando, o esperando al autobús a la vez que navegas. Generalmente los contenidos consumidos a través de un dispositivo móvil se pueden definir con una palabra: "inmediatez". Teniendo en cuenta este concepto existen una serie de características que los dispositivos móviles presentan y que son dignos de mención.

2.4.1. Principales características a tener en cuenta.

2.4.1.1. Tamaño de la pantalla.

Generalmente al navegar a través de un dispositivo móvil dadas las reducidas dimensiones de la pantalla la tendencia será a consumir menor cantidad de información que en un ordenador portátil o de escritorio. Por este motivo es bastante típico que a fin de proporcionar una navegación más satisfactoria, aquellos agentes que publican contenido web, tengan una versión móvil de su página web con un contenido más reducido o con menos criterios de búsqueda de tal modo que la navegación sea más simple y la información más concreta.

2.4.1.2. Localización por coordenadas.

Posiblemente una de las características más importante de los dispositivos móviles es la posibilidad de localizar las coordenadas donde se encuentra el terminal. Esto puede permitir construir aplicaciones basadas en esta característica, como por ejemplo la página web de una inmobiliaria que presente una lista con propiedades a la venta en la zona donde se encuentre el usuario o servicios de redes sociales en los que se puede mostrar los usuarios que están cercanos al terminal y por ejemplo filtrarlos de acuerdo a sus gustos comunes.

Es importante destacar que hasta hace poco tiempo, los servicios de posicionamiento eran provistos por proveedores externos, los cuales posicionaban el terminal a través del punto de acceso que el terminal empleaba para conectarse a la red. Estos servicios tienen dos graves inconvenientes; por un lado son servicios de pago y por otro lado son servicios limitados contractualmente a una cierta localización

geográfica. Esto implica que si una empresa o institución quiere lanzar un servicio basado en posicionamiento en diversos países, ha de tener un proveedor de posicionamiento para cada uno de ellos, pudiendo incluso aparecer condicionamientos de tipo legal. En determinados países la posición en la que se encuentra un usuario se considera como información privada de acuerdo a la legislación. El resultado de estos dos problemas, es que tanto por el precio como por la diversidad de proveedores, puede resultar ciertamente complicado lanzar y mantener un servicio de este tipo a nivel internacional.

Afortunadamente hoy en día los navegadores móviles de última generación son capaces de calcular la posición del terminal accediendo al GPS del mismo lo cual significa un gran avance para los servicios basados en la posición, ya que al no necesitar la presencia de proveedores externos, se reducen costes y se toma el control en cómo gestionar los elementos de índole legal.

2.4.1.3. Ancho de banda.

Tradicionalmente el ancho de banda para la transmisión de datos a través de dispositivos móviles ha sido reducido, y por ello costoso, tanto económicamente, como temporalmente, ya que la descarga de archivos de cierto tamaño requiere mayor tiempo de lo que un usuario está acostumbrado a esperar. Por este motivo los proveedores de contenido tienden a crear versiones reducidas de sus sitios web para la navegación con dispositivos móviles. Este tipo de versiones tienen menos contenido multimedia, y por ejemplo las fotos existentes suelen estar redimensionadas de tal modo que su peso en kilobytes sea menor.

2.4.1.4. Pantalla táctil.

Cuando un usuario visualiza una página web con un ordenador de sobremesa está acostumbrado al empleo del ratón como complemento del teclado para desplazarse a través de la página web que se está visitando. Un buen número de móviles que hoy en día podríamos definir como anticuados, intentaron emular la funcionalidad del ratón a través de cursores o pequeñas almohadillas. Estos intentos se han visto sobrepasados con la aparición de las pantallas táctiles, las cuales suelen ocupar toda la superficie del terminal y permiten la interacción con el usuario en toda la extensión de la misma. Esta funcionalidad hace que en lugar de tener el cursor del ratón como referencia a la hora de interoperar con la pantalla podamos emplear todos y cada uno de los dedos, arrastrando

objetos, aumentando o disminuyendo el nivel de zoom de un mapa o combinando diversas acciones que un simple ratón no permite.

2.4.1.5. Brújula / GPS.

Tal y como definimos anteriormente, la posibilidad de conocer la localización del terminal facilita el desarrollo de aplicaciones de funcionalidad avanzada. Un gran surtido de terminales presenta brújula y navegador GPS que permiten no sólo calcular la posición del terminal sino también indicar al usuario la dirección a seguir. Estos dispositivos hacen que el usuario en lugar de tener que acarrear dos dispositivos (teléfono y GPS o teléfono y mapa) pueda satisfacer sus necesidades con uno sólo.

2.4.1.6. Acelerómetro.

Un acelerómetro es un dispositivo que permite detectar los giros y movimientos de un dispositivo, en este caso un móvil. Si bien esta característica presente en ciertos dispositivos de nueva generación no ha sido empleada por un número masivo de aplicaciones, se puede constatar que ha tenido gran aceptación en el mundo de los juegos para dispositivos móviles y podría extenderse su uso a otro tipo de aplicaciones que permitieran mejorar la calidad de las aplicaciones para dichos terminales.

2.4.1.7. Cámara.

Desde que el empleo de los dispositivos móviles se extendió al gran público una de las funcionalidades añadidas de los terminales ha sido la existencia de una cámara fotográfica y/o de video de mayor o menor calidad. Con este dispositivo, el usuario, más allá de ceñirse al envío de mensajes de texto o la realización de llamadas de voz podía tomar fotografías o videos. Si a ello le añadimos la posibilidad de acceder a la web a través del terminal podemos ofrecer un amplio abanico de posibilidades al usuario. Por ejemplo, será posible inmortalizar aquellos momentos especiales para el usuario a través de un video o una foto. En poco tiempo tenerlos alojados en su red social favorita o en su página web personal, pudiendo incluso indicar en qué lugar la fotografía fue tomada, dado que el terminal conoce la posición del usuario.

2.4.2. Exploradores web para móviles.

En el apartado anterior describimos las características específicas de los dispositivos móviles en contraprestación a aquellas que presenta un dispositivo convencional de sobremesa. Dichas características se basan en el hardware específico de dichos dispositivos, pero es necesario describir que no sólo el hardware difiere cuando hablamos de dispositivos móviles, sino que el software ciertamente también lo hace, siendo especialmente relevante el caso de los navegadores web que emplean dichos terminales.

Los exploradores web para dispositivos móviles han ido evolucionando históricamente, pasando de ser una versión reducida del explorador de sobremesa a ser una herramienta independiente para aprovechar todas las particularidades del entorno en que son usadas.

Existen diversos tipos de exploradores web para dispositivos móviles, un gran grupo de ellos son herramientas que vienen instaladas por defecto en el terminal y son cercanas al sistema operativo, mientras que otras son totalmente externas a él y en ciertas ocasiones precisan de una plataforma intermedia para poder ser ejecutados, como es el caso de los navegadores basados en Java, que necesitan de la máquina virtual de Java para poder ser ejecutados.

Algunas de las características que presentan los navegadores web para dispositivos móviles son:

- Localización geográfica: El navegador puede acceder al GPS del terminal y devolver las coordenadas, la altitud y la precisión de dichos datos.
- Soporte para *add-ons*: Posibilidad de ampliar la funcionalidad del navegador con módulos específicos.
- Soporte para HTML5: En el momento de escribir estas líneas, el nuevo estándar HTML conocido como HTML5 está en pleno proceso de aprobación. Ciertos navegadores han implementado la nueva funcionalidad presente en dicho estándar y son capaces de interpretar las nuevas etiquetas HTML5 como por ejemplo el elemento "canvas" para la representación de elementos gráficos o el elemento "video" para la inclusión de clips de video.
- Optimización para procesadores ARM: Los procesadores de arquitectura ARM *(Advanced RISC machine)* copan el mercado de los dispositivos móviles (98% a fecha de 2007) y por ello los exploradores para dichos terminales están programados para optimizar su rendimiento de acuerdo a esta arquitectura.

• Sincronización de información personal: La "navegación global" es un concepto que han perseguido los desarrolladores de navegadores web que producen versiones de escritorio y para dispositivos móviles. La idea es que el historial, los favoritos, las contraseñas y demás datos que favorecen la navegación personal sean comunes tanto en la versión de escritorio del navegador como en la versión móvil, de tal modo que el usuario tenga la sensación de estar trabajando contra una única aplicación.

2.5 Tecnologías para implementar la solución deseada.

2.5.1. El acceso a la información.

Una de las particularidades más importante de los dispositivos móviles es que los recursos disponibles para el procesamiento del terminal son muy limitados. Por este motivo es bastante habitual encontrarse arquitecturas de tipo cliente-servidor, en las cuales se tienen clientes lo más ligeros posibles que realizan peticiones al servidor a fin de implementar las operaciones deseadas. El envío de peticiones a un servidor en este entorno presenta un problema añadido; no existirá una conexión persistente a la red donde se encuentre el servidor, sino que el dispositivo se conectará a distintos puntos de acceso dependiendo de su localización. En los siguientes apartados se enumeran de acuerdo a su aparición histórica las tecnologías más usuales para la transmisión de datos para dispositivos móviles.

2.5.1.1. GSM.

La tecnología GSM [8] es definida por el consorcio GSM World [9] como "El sistema global para comunicaciones de telefonía móvil. Es la segunda generación de tecnología digital originalmente desarrollada para Europa, pero que en la actualidad acapara el 71 por ciento del mercado mundial. Inicialmente desarrollada para operar en el ancho de banda de 900 MHz y posteriormente modificada para operar con frecuencias de 850, 1800 y 1900 MHzs.

Las siglas GSM provienen de *Groupe Speciale Mobile,* el comité de la Conferencia Europea de Administraciones de Correos y Telecomunicaciones que comenzó el proceso de estandarización de la tecnología GSM. Coloquialmente podemos definir la tecnología GSM como la primera definición del soporte que permite la transferencia de

datos entre dispositivos móviles. El modo de realizar este transporte lo definiremos a continuación.

2.5.1.2. GPRS.

El GPRS (*General Packet Radio Service*) es la primera implementación estandarizada para el intercambio de paquetes dentro la tecnología GSM. GPRS ofrece teóricamente velocidades de transmisión de datos de hasta 115kbits/s, usando técnicas de modularización.

GPRS es el predecesor de 3G dado que el concepto de intercambio de paquetes se aplica del mismo modo en las evoluciones posteriores de esta implementación.

2.5.1.3. EDGE.

La implementación EDGE (*Enhanced Data rates for GSM Evolution*) es definida por GSM World como la última y definitiva etapa en la evolución del estándar GSM. EDGE emplea un nuevo esquema de modularización que teóricamente permite una velocidad de transmisión de datos de hasta 384 kbit/s dentro del espectro GSM.

Es una alternativa en el proceso de actualización hacia la tecnología 3G sin tener que emplear un nuevo espectro. También se la conoce como E-GPRS (*Enhanced GPRS*)

2.5.1.4. UMTS.

El estándar GSM presenta ciertos problemas, tanto de velocidad en la transmisión de datos, como en su internalización (mientras que GSM transmite datos entre las frecuencias 900 MHz y 1800 Mhz, en Estados Unidos se emplean frecuencias por encima de los 1900 MHz). Debido a estos motivos, la Unión Internacional de Telecomunicaciones se encargó de realizar la especificación de los requisitos que debería seguir la siguiente evolución del estándar para las comunicaciones inalámbricas. El trabajo para implementar un estándar que cumpliese estos requisitos fue llevado a cabo por el 3GPP [10] (*The 3rd Generation Partnership Project*), y el producto de su trabajo es la tecnología conocida como UMTS (*Universal Mobile Telecommunications System*).

UMTS, a veces referido como 3GSM a fin de remarcar la evolución desde GSM, es el termino que agrupa a todas las tecnologías inalámbricas desarrolladas dentro del proyecto 3GPP. Estas tecnologías están basadas en la tecnología W-CDMA que también era la base para las tecnologías GSM.

Las implementaciones de UMTS ofrecen una velocidad de un máximo de 2Mbit/s para usuarios que estén en movimiento o quietos, y 348 kbit/s en vehículos en movimiento, siendo posible desarrollar aplicaciones que precisen de un mayor ancho de banda.

2.5.2. Sistemas operativos.

2.5.2.1. Symbian.

Symbian es un sistema operativo especialmente diseñado para dispositivos móviles. En un principio fue creado por un consorcio de empresas de telefonía móvil a fin de competir con los sistemas operativos de Palm y Microsoft (Windows Mobile). En la actualidad la fundación Symbian pertenece en su totalidad a Nokia y es el sistema operativo más extendido en los terminales de dicha marca.

Una de las principales ventajas del desarrollo de aplicaciones para este sistema operativo es que los terminales Symbian son mayoría en el mercado (alrededor del 40 %), por lo que el desarrollo de aplicaciones para este sistema operativo posibilitará el acceso a un gran número de usuarios. Al mismo tiempo, Symbian, desde su nacimiento, ha sido un sistema operativo ideado para dispositivos móviles en contrapunto a otros sistemas presentes en el mercado.

2.5.2.2. Windows CE.

Una de las principales desventajas del desarrollo de aplicaciones para este sistema operativo es que Windows CE pertenece a la familia Windows y por ello tiene la fuerte herencia de un sistema operativo que en su origen no fue diseñado para aplicaciones móviles. Ciertamente ante un terminal con sistema operativo Windows CE, se puede tener la sensación de que se está ante una versión “comprimida” del sistema operativo diseñado para terminales de sobremesa.

Una clara ventaja de desarrollar aplicaciones para Windows CE es la facilidad de integración con otros productos de la familia Microsoft, tales como Microsoft Office, el uso de los cuales está muy extendido en el mercado.

2.5.2.3. Linux.

La familia de sistemas operativos Linux también presenta distribuciones especialmente diseñadas para su implantación en dispositivos móviles.

En la primera época del desarrollo de las aplicaciones para móviles, cada dispositivo precisaba de un sistema operativo específicamente desarrollado para su hardware. Esta situación hacía el desarrollo bastante costoso, y por ello se empezó a trabajar en soluciones más abstractas, muchas de ellas basadas en distribuciones Linux ya existentes.

Una distribución Linux básica ocupa en torno a 2 megabytes, lo cual la hace especialmente atractiva debido a los limitados recursos de este tipo de dispositivos.

Como resultado de esta corriente de desarrollo nació la EMC *(Embedded Linux Consortium)* o Consorcio de Linux Embebido a fin de desarrollar y promover estándares Linux para aplicaciones móviles.

A la hora de desarrollar aplicaciones de tipo cliente para dispositivos móviles basados en Linux, Java es el lenguaje de programación de mayor uso y éxito dado que la especificación J2ME/MIDP ha sido adaptada como un estándar para la mayoría de fabricantes de dispositivos móviles

2.5.2.4. Android.

Dentro de los sistemas operativos basados en Linux dedicamos un especial apartado a Android, sobre todo por haber sido promovido por una de las grandes compañías del sector, Google.

Android está basado en un kernel Linux 2.6 y presenta un SDK para el desarrollo de aplicaciones en Java. La popularidad de este lenguaje de programación y el apoyo de Google ha hecho que gane en popularidad, y pese a ser un sistema operativo relativamente joven, ya pueden encontrarse en el mercado dispositivos móviles basados en él.

2.5.3. Programación de aplicaciones para dispositivos móviles.

Una vez que las tecnologías de acceso a la red y los distintos sistemas operativos han sido presentadas, pasaremos a mostrar una selección de diversos lenguajes de programación existentes en el mercado para desarrollar aplicaciones. Dichos lenguajes han de tener ciertas características especiales, dado que tal y como comentamos anteriormente, los recursos de los dispositivos móviles son muy limitados, por lo cual los clientes a implementar han de ser muy minuciosos a la hora de tratar recursos como por ejemplo la memoria o el uso del procesador. A continuación mostramos algunos de

los lenguajes de programación de uso más extendido para el desarrollo de aplicaciones móviles.

2.5.3.1. Java ME.

Java Micro Edition es la versión del lenguaje de programación Java orientada a dispositivos móviles. Se define como un subconjunto de las interfaces de Java especialmente aplicables al desarrollo de aplicaciones para móviles.

La implementación de clientes con Java ME presenta ciertas ventajas. Por ejemplo, ciertos entornos de desarrollo para Java ME ofrecen emuladores que permiten probar el comportamiento de los clientes durante el desarrollo, y su instalación en el dispositivo a través de un sencillo instalador una vez probados.

Las aplicaciones implementadas en Java ME presentan una configuración especial, de tal modo que tanto la JVM (Máquina Virtual de Java) como las bibliotecas necesarias se minimizan para optimizar los recursos del terminal. El estándar J2ME/MIDP proporciona a los desarrolladores interfaces especialmente diseñadas para optimizar los recursos de las aplicaciones desarrolladas para dispositivos móviles.

Teniendo en cuenta que la mayoría de fabricantes incluyen el estándar J2ME/MIDP en todos sus terminales, los desarrolladores pueden escribir aplicaciones disponibles para casi todos los terminales del mercado.

2.5.3.2. Programación en Symbian.

Tal y como explicamos anteriormente, el sistema operativo Symbian es el propio de los dispositivos móviles fabricados por Nokia.

El sistema operativo Symbian ofrece ciertas interfaces de usuario para la programación de aplicaciones, de ellas la más conocida es la plataforma S60, la cual es un conjunto de aplicaciones y librerías basadas en C++ que permiten al desarrollador la programación de aplicaciones para aquellos dispositivos que tengan Symbian como sistema operativo.

2.5.3.3. .Net para móviles.

Del mismo modo que Microsoft tiene un entorno de desarrollo para sus aplicaciones de escritorio, posee un entorno para el desarrollo de aplicaciones orientadas a los dispositivos móviles. El entorno de desarrollo de Microsoft conocido como Visual

Studio dispone de una extensión llamada SDE *(Smart devices extensión)* para la creación de dichas aplicaciones, tanto en Visual Basic .Net como en C# .Net.

Para poder ejecutar una aplicación desarrollada en dicho entorno, el dispositivo móvil ha de tener instalado el *.Net Compact Framework*, más ligero que la versión para aplicaciones de sobremesa, pero necesario para que el dispositivo pueda entender las instrucciones programadas en Visual Studio .Net.

Lógicamente al estar el entorno de desarrollo integrado dentro de la plataforma .NET, dispondremos de interfaces para comunicarnos con otras aplicaciones desarrolladas en el mismo entorno, tales como las versiones móviles de la suite Microsoft Office, y librerías para gestionar los recursos específicos de un dispositivo móvil, por ejemplo una pantalla táctil.

La ventaja de desarrollar aplicaciones empleando Microsoft Visual Studio .NET y la extensión para el desarrollo de aplicaciones para dispositivos móviles es el hecho de que muchos desarrolladores estén familiarizados con este entorno de desarrollo. Así mismo al compartir los fundamentos del *.Net Framework*, ciertas aplicaciones que en un principio se diseñaron como aplicaciones de escritorio de servidor, pueden ser migradas fácilmente al entorno de ejecución de un dispositivo móvil.

Al mismo tiempo, la desventaja de este entorno es el de poder ejecutarse sólo en dispositivos que tenga un sistema operativo de la familia Windows CE, junto al hecho de ser un producto propietario, tanto a la hora del desarrollo como a la de comprar el terminal (La licencia de Windows CE no es gratuita).

2.5.3.4. La opción del navegador web (XHTML).

La mayoría de navegadores web comerciales existentes en el mercado presentan versiones especialmente adaptadas o son compatibles con tanto el tamaño de pantalla como las posibilidades de interacción con el usuario de los dispositivos móviles.

Ciertamente una opción a la hora de implementar una arquitectura cliente-servidor es en lugar de tener un cliente programado en un lenguaje de programación específico como Java o Symbian, enviar una petición http a través del explorador web y mostrar el resultado de la petición a través del explorador en conformidad con la especificación XHTML.

La especificación XHTML [11] es una evolución de HTML *(HyperText Markup Language)*. El HTML es el estándar para publicar hipertexto en la web. Es un formato

no propietario basado en SGML, pudiendo ser creado y procesado por un gran número de herramientas, desde simple editores de texto, hasta herramientas de tipo gráfico. HTML emplea etiquetas tales como <h1> y </h1> para estructurar el texto dentro de encabezados, párrafos, listas, enlaces de hipertexto, etc.

El estándar XHTML *(Extensible HyperText Markup Language)* es una familia de tipos de documentos y módulos, actuales y futuros que reproducen, reducen y extienden HTML, de tal modo que al realizarse esta reformulación, la base del estándar sea XML (eXtensible Markup Language) y no SGML como en su predecesor. Los tipos de documentos de la familia XHTML están todos basados en XML y están diseñados para funcionar con servicios o clientes basados en XML.

2.5.4. JavaScript.

Tal y como presentamos en el subapartado anterior, existen diversos navegadores web en el mercado especialmente optimizados para el consumo de información en un dispositivo móvil. No obstante, el modo tradicional de entender la arquitectura cliente – servidor en este contexto implica la creación de una página web en formato HTML o XHTML en un servidor web. Esta página se envía al cliente, en este caso un navegador web, el cual muestra dicha página al usuario, pero la interacción del usuario con la página recibida se entiende como más bien escasa, ya que básicamente, se puede leer o escuchar la información obtenida y continuar la navegación mandando una nueva petición al servidor. Es decir, el navegador web actúa como soporte para una información estática que se define a nivel de servidor y que es consumida por el usuario, el cual no puede interactuar con esta información más allá de la simple visualización de la misma. A fin de romper esta limitación y mejorar la interacción con la información proveniente del servidor apareció JavaScript.

Podemos definir JavaScript como un lenguaje de programación interpretado que pese a poderse ejecutar en el lado del servidor *(Server Side JavaScript)* presenta un mayor uso en aplicaciones de tipo cliente. En nuestro caso particular nos interesa por su presencia en la gran mayoría de navegadores web existentes en el mercado. Históricamente fue desarrollado por Netscape, empresa que posteriormente fue adquirida por American Online. Dado el gran éxito que tuvo en el mercado el navegador Netscape Navigator 3.0, el uso de JavaScript se fue popularizando e incluso Microsoft desarrolló una solución muy parecida, por no llamarla copia, llamada JScript. No obstante Netscape decidió formalizar el lenguaje y por ello se desarrolló el estándar

ECMAScript por parte de la ECMA (*European Computer Manufacturers Association).* Las versiones que los navegadores comerciales presentan hoy en día no son más que implementaciones de dicho estándar . [12]

La gran ventaja de la presencia de JavaScript en los navegadores web es la posibilidad de que el contenido de las páginas web sea dinámico, es decir, que dependiendo de la interacción que el usuario tenga con la página, esta presentará un comportamiento u otro. Esto es posible dado que JavaScript puede acceder y manipular el DOM (*Document Object Model* o Modelo de Objetos del Documento). El DOM es una interfaz de programación para poder manipular los objetos que representan un documento HTML o XML. A través de esta interfaz podemos eliminar, añadir o modificar elementos de un documento XHTML en función del comportamiento del usuario, lo que nos permite mayor versatilidad, mayor interoperabilidad y sobre todo minimizar el número de comunicaciones con el servidor. Un claro ejemplo del uso exitoso de JavaScript lo tenemos en las comunicaciones de tipo asíncrono o AJAX, por las cuales es posible enviar información entre el cliente y el servidor sin necesidad de recargar toda la página. AJAX ha revolucionado en este sentido la web tal y como la conocíamos dado que, con anterioridad, cualquier tipo de comunicación que quisiese enviarse desde el cliente al servidor web precisaba de una recarga de la totalidad de la página.

JavaScript es un buen complemento de XHTML desde el punto de vista de que la estandarización y claridad del segundo se compaginan con la versatilidad y dinamismo del primero. Existe una amplia cantidad de librerías y aplicaciones programadas en JavaScript que hacen realidad el concepto de navegación amigable para el usuario.

Las primeras implementaciones de JavaScript en los dispositivos móviles las podemos definir como algo pesadas y un poco alejadas de la perfección, pero en los terminales de última generación la mayor cantidad de recursos hardware han hecho posible que la ejecución de JavaScript sea más fiable y precisa.

2.5.5. El estándar WAP.

Con anterioridad a la aparición de XHTML, se creó un estándar para definir las comunicaciones entre servidores de aplicaciones de valor añadido para dispositivos en inalámbricos y dichos dispositivos. Este estándar, denominado *WAP (Wireless Aplication Protocol)*, no permitía en su primera versión la navegación por Internet, sino

simplemente el consumo de servicios alojados en la red que estuviesen implementados en el lenguaje *WML (Wireless Markup Language)*.

El lenguaje *WML* es un lenguaje de presentación de contenidos que permite consumir menos recursos del terminal que *HTML* a la par que presenta funciones específicas para dispositivos con una pantalla de tamaño reducido.

El primer estándar WAP presentaba una pila de protocolos similar a la pila TCP/IP sobre la que se fundamente Internet, pero incompatible con esta. Por este motivo era necesario que las aplicaciones WAP se comunicaran con los servidores HTTP a través de pasarelas WAP que resolvieran esta incompatibilidad. Este problema se solucionó al aparecer el estándar WAP 2.0, en el cual se incluyeron los protocolos necesarios para solventar esta incompatibilidad. No obstante las pasarelas WAP siguen existiendo por motivos de eficiencia, tales como el almacenamiento de información en cache por poner un ejemplo.

Uno de los usos más extendidos del estándar WAP es a través del empleo de mensajes de tipo *WAP Push*. Este tipo de mensaje consiste en uno o varios mensajes SMS que contienen código WML en el cual se encuentran enlaces a uno o más recursos externos, tales como aplicaciones escritas en Java ME, contenidos multimedia, etc. El terminal al recibir estos mensajes es capaz de interpretar su contenido y acceder a dicho recurso. De este modo es bastante fácil comunicar al usuario final donde localizar el contenido que está interesado en adquirir.

La implantación del estándar WAP ha sido ciertamente lento debido a las pasarelas WAP. Estas pasarelas son controladas por las operadoras de telefonía con las cuales los proveedores de contenidos han tenido que negociar. Este costoso proceso ha tenido como resultado que en el mercado se puede apreciar una mayor existencia de aplicaciones web basadas en XHTML que aplicaciones WAP, pese a ser la segunda tecnología anterior a la primera.

2.5.6. La solución a implementar.

En los últimos años hemos podido observar como el sector de la telefonía móvil ha evolucionado con la aparición de las redes de tercera generación, las cuales proporcionan un ancho de banda de hasta 2 megabytes por segundo. Los terminales han pasado de ser un dispositivo para realizar llamadas telefónicas, enviar mensajes, realizar fotografías o escuchar música, a ser empleados para navegar por Internet.

El usuario medio es más favorable a consultar información de modo puntual, pero cada vez más reacio a descargarse aplicaciones que tal vez sólo vaya a emplear ocasionalmente y que pueden ser costosas desde el punto de vista económico, dado que muchos operadores continúan facturando por volumen de datos descargado. De este modo parece claro que las aplicaciones del tipo JAVA ME han quedado obsoletas para nuestro propósito, debido a la necesidad de ser descargadas en su totalidad.

En lo que concierne a las aplicaciones desarrolladas para un sistema operativo concreto existen ventajas y desventajas. La gran ventaja es el ser altamente eficientes por estar desarrolladas en lenguajes optimizados para entornos concretos, pero por el contrario dichas aplicaciones no serían válidas para otros sistemas operativos. Pongamos que eligiésemos desarrollar nuestra aplicación para dispositivos de la familia Symbian. Posiblemente la aplicación funcionaría de un modo satisfactorio para dichos dispositivos, pero dejaríamos de lado el concepto de universalidad ya que la aplicación no funcionaría en terminales con otro sistema operativo. Si decidiésemos ofrecer nuestra aplicación para todos los sistemas operativos presentes en el mercado, los costes de desarrollo se dispararían, ya que se necesitaría implementar la aplicación en Symbian, Android, iOS, Windows mobile, etc

Considerando las posibilidades de navegación web en este tipo de dispositivos existentes en la actualidad, parece que una presentación en una página web especialmente diseñada para un móvil o una PDA pueden proporcionar al usuario una experiencia tan rica como la que se tiene en un computador convencional, sobre todo al poder emplear JavaScript para facilitar la interacción entre el usuario y la aplicación. Al mismo tiempo consideramos que las redes de tipo GSM, EDGE y UMTS disponibles en la actualidad, proveen la velocidad suficiente para que el usuario disfrute de una navegación web aceptable.

Tal y como explicamos con anterioridad, la especificación XHTML del consorcio World Wide Web es una evolución del HTML tradicional para definir documentos que puedan ser visionados en la web. Cualquiera de los navegadores web presentes en un dispositivo móvil no demasiado antiguo puede mostrar páginas web en formato XHTML. Este será el formato que emplearemos para presentar la información requerida por el usuario. No obstante a la hora de implementar la solución se nos plantea una pregunta en torno a cómo agrupar un conjunto de páginas XHTML sin que la solución parezca una amalgama de código. A fin de dotar a la aplicación de orden, limpieza y funcionalidad construiremos un gestor de contenidos en el lenguaje de programación

PHP, el cual producirá nuestras páginas XHTML. Antes de continuar, presentaremos lo que es un gestor de contenidos y el lenguaje de programación PHP a fin de justificar el motivo de nuestra decisión.

2.5.6.1. Gestor de contenido o CMS *(Content Management System).*

Un gestor de contenido es una aplicación diseñada para crear, dar formato, reproducir y gestionar información en un entorno Web. En la mayoría de la ocasiones están programados con lenguajes de programación de tipo script y están alojados en el servidor web, de tal forma que se puedan acceder directamente a recursos del servidor tales como bases de datos, recursos multimedia o servicios web.

Ciertamente el concepto de gestor de contenido es bastante amplio ya que este tipo de aplicaciones pueden ser usadas en ámbitos diversos tales como redes internas de empresas para presentar información interna, sitios web que presenten contenidos de acceso libre para usuarios anónimos, etc. No obstante en nuestro caso y dada la naturaleza de los dispositivos con los que vamos a utilizar la aplicación nos interesan especialmente los gestores de contenido para dispositivos móviles.

Un gestor de contenido para dispositivos móviles o MCMS (del término inglés *Mobile Content Management System)* es una aplicación que es capaz de proveer de contenidos a un dispositivo móvil. Los MCMS pueden ser aplicaciones independientes o extensiones de un CMS genérico de tal modo que se tengan en cuenta las peculiaridades de dichos dispositivos. Entre estas características especiales se encuentran por ejemplo, el tamaño de pantalla, las limitaciones en la conectividad, la escasa capacidad de almacenamiento en memoria, la menor capacidad de procesamiento en comparación con un ordenador personal, así como las posibilidades de conocer la posición del dispositivo.

2.5.6.2. El lenguaje de programación PHP.

Tal y como describimos anteriormente, la solución a implementar estará basada en un gestor de contenido escrito en el lenguaje de programación PHP.

De acuerdo a la definición presentada en la página web oficial del lenguaje, PHP es un lenguaje de tipo script interpretado en el lado del servidor, de ámbito general y de amplio uso, especialmente adecuado para desarrollo Web y que puede estar embebido dentro de código HTML. De acuerdo con el estudio de Netcraft Survey de abril de 2007 es usado por 20 917 850 dominios y 1 224 183 direcciones IP únicas.

PHP fue originado en 1994 como un proyecto personal de Rasmus Lerdorf para contabilizar el número de personas que visitaban su curriculum vitae en la red. Al ver que un creciente número de personas se interesaban por este sistema, lo público dándole el nombre de Personal Home Page Tools a mediados de 1995. Esta aplicación incluía un parser que posibilitaba interpretar archivos en un rudimentario PHP. Tras esta primera versión el lenguaje creció y otra serie de desarrolladores se incorporaron al proyecto. A mediados de 1997 existían cerca de 50.000 sitios web creados con PHP, creciendo la popularidad del lenguaje especialmente con su tercera versión la cual integraba conectividad con Mysql. Con la publicación de PHP 4 se aumentó la funcionalidad del lenguaje y con la versión 5 se aumentó su efectividad.

Pese a dominar el mercado PHP tiene competidores, pudiendo considerarse a sus máximos concurrentes ASP de Microsoft o JSP de Oracle.

3. GESTIÓN DEL PROYECTO

Dado que la naturaleza del presente proyecto es modular, descompondremos la gestión del mismo en una serie de tareas en función de los módulos que componen el mismo, asignando para cada tarea recursos tanto económicos como temporales. El resultado de este estudio ha de ser la planificación temporal del proyecto junto a una estimación de los costes materiales del mismo.

A. Definición de tareas.

En este apartado definiremos las tareas a realizar durante el proyecto, especificando la duración, los recursos asignados y los objetivos de las mismas

A.1. Estudio del Arte

Descripción: Estudio general sobre la actividad del sector turístico. Estudio sobre las soluciones turísticas existentes en el mercado para dispositivos móviles. Estudio tecnológico.

Objetivos: Conocer el sector para el cual se va a desarrollar el proyecto fin de carrera, así como las aplicaciones existentes para dispositivos móviles. Conocer las tecnologías existentes en el mercado para implementar la solución deseada.

Duración: 2 semanas

Esfuerzo: 1 persona / mes

A.2. Análisis y diseño de alto nivel.

Descripción: Análisis global del proyecto SAMAP. Diseño de alto nivel para la construcción de un módulo gestor de comunicaciones y una aplicación de cliente de tipo CMS.

Objetivos: Conocer las particularidades del proyecto SAMAP y de los módulos ya existentes o en fase de implementación. Realizar el diseño a alto nivel de una aplicación web como aplicación de cliente para la generación de itinerarios turísticos. Realizar el diseño a alto nivel de un módulo gestor de comunicaciones entre los módulos existentes en el sistema.

Duración: 1 semana

Esfuerzo: 1 persona / mes

A.3. Análisis y diseño a bajo nivel del gestor de contenidos

Descripción: Análisis de la funcionalidad y diseño del motor del gestor de contenido. Definición de componentes, páginas, javaScript, estilos, plantillas y bibliotecas auxiliares.

Objetivos: Definir y delimitar la funcionalidad del gestor de contenidos y sus partes. Definir los flujos para crear páginas XHTML y modular cómo los distintos elementos interaccionan para lograr tal fin.

Duración: 1,5 semanas

Esfuerzo: 1 persona/mes

A.4. Integración con el servidor de puntos de interés

Descripción: Análisis del proyecto de fin de carrera de Israel Barroso Pérez "Extracción automática de información turística de Internet.". Análisis de las interfaces de comunicación del mismo. Implementación del componente POI en el CMS.

Objetivos: Implementar un componente para el gestor de contenidos a fin de poder consumir las interfaces de búsqueda de puntos de interés turístico que ofrece el servidor de puntos de interés turístico.

Duración: 2 semanas

Esfuerzo: 1 persona/mes

A.5. Integración con el servidor de mapas

Descripción: Análisis del proyecto de fin de carrera de Alberto Anta Andrés "Implementación de un módulo de gestión de mapas utilizando herramientas de software libre". Análisis de las interfaces de comunicación del mismo. Implementación del componente mapas en el CMS.

Objetivos: Implementar un componente para el gestor de contenidos a fin de poder consumir las interfaces de recuperación de coordenadas y mapas que ofrece el servidor de mapas.

Duración: 1 semana

Esfuerzo: 1 persona/mes

A.6. Integración con el servidor de itinerarios turísticos.

Descripción: Análisis y definición de las interfaces de comunicación con el servidor de itinerarios turísticos.

Objetivos: Implementar un componente para el gestor de contenidos a fin de poder consumir la interfaz que el servidor de generación de itinerarios turísticos ofrece. Definir esta interfaz dado que el servidor no ha sido desarrollado en el momento de realización del presente proyecto.
Duración: 0,5 semanas.
Esfuerzo: 1 persona/mes

A.7. Implementación de la aplicación web.

Descripción: Implementación de la aplicación web usando el gestor de contenido y sus componentes, de tal modo que los usuarios de la misma puedan generar itinerarios turísticos de acuerdo a sus preferencias.
Objetivos: Implementar y probar la aplicación web y su módulo de comunicaciones a fin de facilitar al usuario la generación de itinerarios de interés turístico y su consumo en el dispositivo móvil.
Duración: 2 semanas.
Esfuerzo: 1 persona/mes

A.8. Documentación del sistema.

Descripción: Generación de la documentación del sistema. Documentación del código. Escritura de la memoria y de los manuales de usuario/administración.
Objetivo: Obtener un documento con el especificación del desarrollo del presente proyecto de fin de carrera. Obtener un manual de usuario/administrador a fin de que el sistema pueda ser empleado en otros contextos.
Duración: 1,5 semanas.
Esfuerzo: 1 persona/mes

A.9. Pruebas y conclusiones.

Descripción: Pruebas para determinar que el sistema funciona de acuerdo a las especificaciones. Conclusiones sobre los resultados del proyecto de fin de carrera.
Objetivo: Comprobar que la aplicación web construida sobre el CMS satisface los objetivos del presente proyecto. Generar un documento con las conclusiones de la elaboración del proyecto de fin de carrera.
Duración: 0,5 semanas.
Esfuerzo: 1 persona/mes

B. Recursos

Para la realización del presente proyecto fin de carrera necesitaremos una serie de recursos que a continuación pasamos a citar.

B.1. Infraestructura.

Para la realización de las tareas anteriormente citadas necesitaremos disponer de un local de trabajo en el cuál, a modo de oficina, sentarse a trabajar. Cotejando los precios del mercado, un puesto de oficina con acceso a internet de banda ancha e incluyendo servicios de limpieza supone unos costes de 250 € al mes.

Los recursos materiales a consumir suman un coste total de 250 € al mes.

B.2. Recursos materiales.

Para del desarrollo del presente proyecto se utilizará sólo una computadora portátil modelo Acer TravelMate con un procesador Intel® Core Solo CPU U3500 @ 1.40 GHz con 2,92 GB de memoria RAM. Este modelo en concreto con dos años de antigüedad se encuentra descatalogado. No obstante un modelo de similares características tiene un valor de 798 € en el mercado actual. Por tanto en la actualidad podemos asumir un valor marginal de la máquina de desarrollo de unos 400 €.

Los recursos materiales a consumir suman un coste total de 400 €.

B.3. Recursos de software.

Durante el proyecto se utilizarán herramientas de dos tipos, sistemas operativos y aplicaciones. Todos ellos son libre o bien venían incluidos con la distribución de la máquina de desarrollo, por tanto no incrementan el coste del proyecto en modo alguno.

Sistemas Operativos:

- Windows XP SP 3 – sistema operativo sobre el que corre la aplicación virtual de VMware ®. Incluido con la máquina de desarrollo (0 €)
- Xubuntu 10.04 – sistema operativo de la máquina virtual de VMware ® sobre la que se desarrollará la aplicación (0 €)

Herramientas

- Paquete ofimático Microsoft ® Office 2007 – Se empleará para la redacción de la memoria del proyecto. Incluido con la máquina de desarrollo (0 €)
- VMware Player – Herramienta para la ejecución de máquinas virtuales. Gratuita (0 €)
- UMLArgo 0.32.2 – Herramienta para la realización de diagramas UML. Gratuita (0 €)
- Dia v.0.97 – Herramienta para la realización de diagramas de flujo. Gratuita (0 €)
- Netbeans 6.8 – Entorno de desarrollo para PHP y Java. Gratuita (0 €)
- PHPDocumentor – Herramienta para la generación de documentación. Gratuita (0 €)

Los recursos de software a consumir suman un coste total de 0 €.

B.4. Recursos humanos

Dada la no existencia de un colegio de ingenieros técnicos en informática en la Comunidad de Madrid es complicado establecer el precio por hora de un analista-programador. Pero en base al conocimiento personal del mercado podemos estimar este precio en unos 40 € a la hora.

C. Resultado de la planificación

La duración del proyecto se estima en 12 semanas (aproximadamente 3 meses) en jornadas de 8 horas durante 5 días por semana. El diagrama de Gantt que se muestra a continuación muestra un desglose de esta planificación.

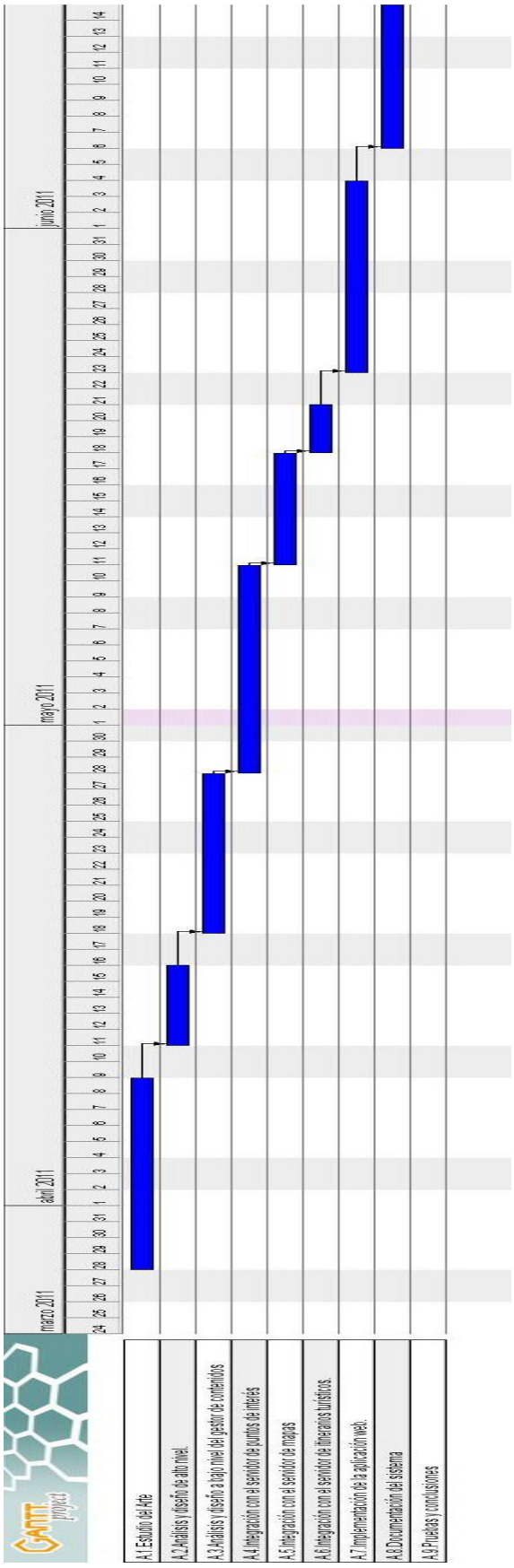

Figura 3 - Diagrama de Gantt

D. Resumen de los costes.

El resumen de los costes derivados de la ejecución del presente proyecto se muestra en una tabla a continuación. Todos los precios aparecen en euros.

Concepto	Cantidad	Precio Unitario	Importe
Infraestructura	3	250	750
Hardware	1	400	400
Software	7	0	0
Recursos Humanos	480	40	19200
Subtotal			20350
(16 % IVA)			3256
TOTAL			**23606**

El coste total del proyecto asciende a VEINTITRES MIL SEISCIENTOS SEIS EUROS

Madrid, 15 de julio de 2011

El Ingeniero Técnico Informático proyectista

Fdo. Víctor Manuel Rincón de Luis.

4. ANÁLISIS DEL PROYECTO

4.1 Visión global.

Tal y cómo se presentó con anterioridad, el presente proyecto se engloba dentro de un proyecto de mayor dimensión, el proyecto SAMAP. Teniendo en cuenta que el objetivo del mismo es la construcción de una aplicación para la creación de itinerarios turísticos es importante definir la estructura del mismo de un modo global. El diseño de alto nivel lo podemos observar en el siguiente gráfico:

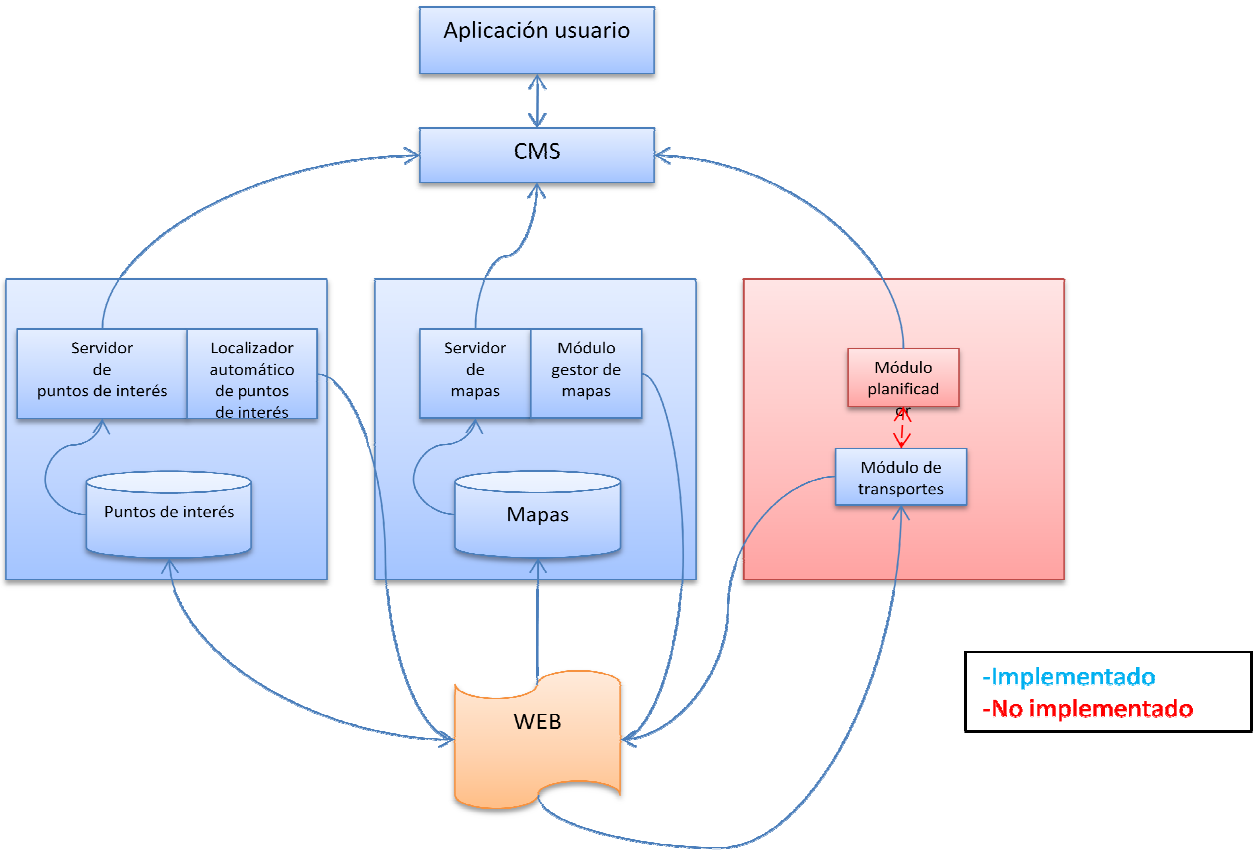

Figura 4 Proyecto SAMAP – Diseño alto nivel

Cómo se puede apreciar en el gráfico, la práctica totalidad del sistema está casi terminada, quedando sólo el módulo planificador y su conexión con el módulo de transportes pendientes por implementar. Tras consultar con la dirección del proyecto, hemos decidido emular el comportamiento de ese módulo, pese a lo cual, diseñaremos una interfaz de comunicación con dicho módulo. Esta interfaz ha de ser lo

suficientemente flexible como para que la persona que en el futuro lleve a cabo la implementación de dicho módulo pueda simplemente acoplarse al conjunto del sistema.

Dentro del gráfico previo la parte que concierne propiamente al presente proyecto es la aplicación de usuario, el CMS y la comunicación con el servidor de puntos de interés, el servidor de mapas y el planificador.

La idea básica es la de separar la funcionalidad en módulos independientes que se comunicarán con el CMS, dejando sobre él tanto la tarea de gestionar el tráfico del sistema y de recoger las peticiones de la aplicación como la de enviar el resultado de dichas peticiones al usuario final. En cierto sentido, lo que haremos será considerar el módulo gestor de comunicaciones como una abstracción que se implementa a través de módulos independientes del CMS. A continuación presentamos las distintas partes a integrar en el proyecto SAMAP:

- **Aplicación web para dispositivos móviles**: Aplicación de tipo cliente que genera una serie de páginas web en código XHTML que podrán ser consumidas por el navegador de un dispositivo móvil. Esta aplicación presentará al usuario la información proveniente del resto de módulos y se encargará de recopilar la información necesaria para la creación del itinerario turístico y el visionado del mismo.
- **Módulo gestor de comunicaciones**: Objetivo central de este proyecto. Gestionará las comunicaciones entre la aplicación de usuario y el resto de módulos.
- **Localizador automático de puntos de interés**. Módulo encargado de rastrear la web para encontrar atracciones turísticas. La información recopilada será almacenada en el sistema a fin de que el servidor de monumentos pueda proporcionar la información deseada al usuario.
- **Servidor de puntos de interés**. Módulo encargado de facilitar información relacionada con atracciones turísticas al módulo gestor de comunicaciones en función de los criterios de búsqueda del usuario, teniendo como fuente de información el almacén de datos originado por el localizador automático de monumentos.
- **Módulo planificador**. Módulo encargado de planificar el itinerario turístico a realizar con las atracciones turísticas seleccionadas. El planificador se comunicará con el módulo de transportes a fin de optimizar el itinerario y elegir los medios de transportes adecuados.
- **Módulo de transportes**. Módulo encargado de proporcionar información sobre cómo ir de un lugar a otro en el destino seleccionado por el usuario de la aplicación.

• **Servidor de mapas**. Módulo que en facilitará la información relativa a los mapas de tal modo que el usuario pueda acceder a los mismo durante el transcurso de su itinerario.

• **Módulo gestor de mapas**. Módulo que en función del itinerario seleccionado por el usuario buscará el mapa del itinerario en la web y lo almacenará en el sistema.

4.2 Especificación de requisitos.

A continuación presentaremos la especificación de requisitos de la aplicación. Para ello hemos optado por emplear UML de un modo simplificado, ya que sólo presentamos aquellos elementos que proporcionan información relevante y obviamos aquellos elementos que no la aportan, a fin de maximizar la claridad del documento.

Comenzaremos presentando los casos de uso que hemos detectado para el sistema, y a continuación mostraremos el diagrama de secuencia de la aplicación.

4.2.1. Casos de uso.

4.2.1.1. Caso de uso: “Crear itinerario turístico”.

Descripción:

Caso de uso de nivel general, en el cual se conectará a la aplicación web a fin de generar un itinerario turístico en su dispositivo móvil.

Nivel:

Alto nivel.

Actor principal:

- Usuario.

Prerrequisitos:

- El usuario ha de disponer de un dispositivo móvil dotado de un explorador web con conexión a internet.

Post-condiciones:

En caso de éxito:

- El usuario dispondrá de un itinerario turístico en su dispositivo móvil que podrá emplear para realizar su visita turística al destino deseado.

En caso de fracaso:

- El usuario no dispondrá de un itinerario turístico en su dispositivo móvil.

Disparador:

El usuario introduce en el navegador web de su dispositivo móvil la página de inicio de la aplicación web.

Escenario principal en caso de éxito:

1. El usuario define los criterios de búsqueda para las atracciones turísticas en las que está interesado.
2. El usuario selecciona las atracciones turísticas que desea visitar.
3. El usuario envía una petición a la aplicación para generar un itinerario turístico.
4. El usuario recibe el itinerario turístico y visualiza los mapas.

Representación gráfica:

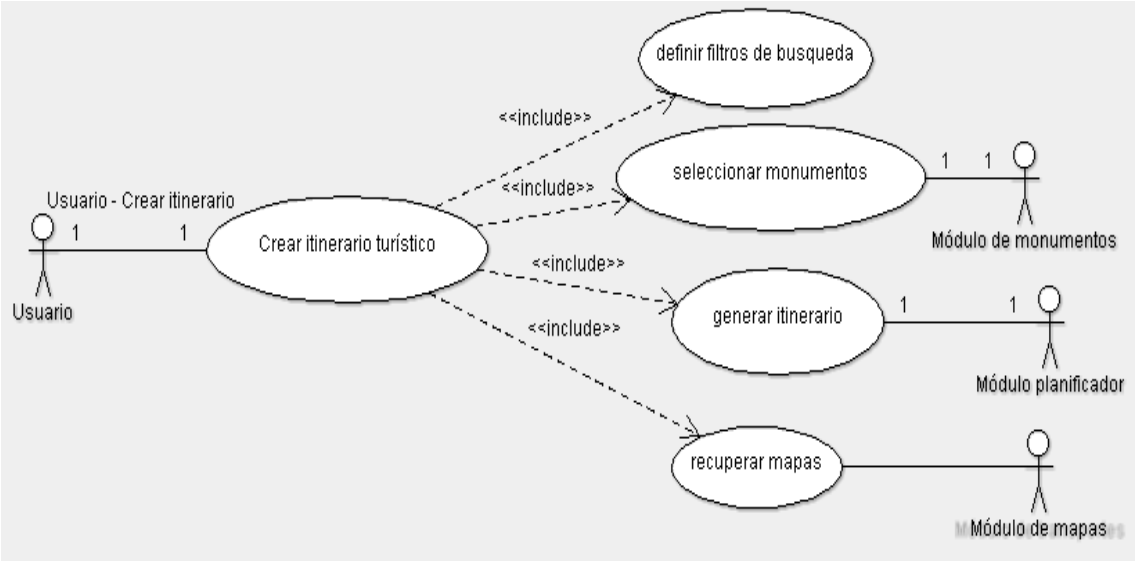

Figura 5 Caso de uso "Crear itinerario turístico"

Frecuencia:

Cada vez que un usuario decida conectarse a la aplicación web.

Suposiciones:

- El usuario tiene suficientes conocimientos sobre cómo navegar por una página web.
- La aplicación se encuentra en estado de ejecución en el servidor.
- La aplicación puede consumir la interfaz del módulo de monumentos.
- La aplicación puede consumir la interfaz del módulo planificador.
- La aplicación puede consumir la interfaz del módulo de mapas.

Requisitos especiales

- Interfaz de usuario

  La interacción con el usuario debe estar adaptada a las peculiaridades de un dispositivo móvil (tamaño de pantalla, juego de caracteres, etc.)

4.2.1.2. Caso de uso: “Definir filtros de búsqueda ”

Descripción:

Caso de uso de nivel medio, a través del cual el usuario definirá cuáles son sus preferencias en los que se refiere a las atracciones turísticas que desea visitar.

Nivel:

Nivel medio.

Actor principal:

- Usuario.

Prerrequisitos:

- El usuario ha de iniciar el proceso de creación de itinerario.

Post-condiciones:

En caso de éxito:

- El sistema conocerá las preferencias del usuario a la hora de realizar búsquedas de atracciones turísticas

En caso de fracaso:

- El sistema desconocerá las preferencias del usuario a la hora de realizar búsquedas de atracciones turísticas y la creación del itinerario se verá interrumpida.

Disparador:

El usuario accede a la página de entrada a la aplicación.

Escenario principal en caso de éxito:

1. El usuario accede a la página de entrada a la aplicación.
2. El usuario rellena los campos del formulario donde se recaba la información acerca de sus preferencias.
3. El usuario envía sus preferencias al servidor.

Representación gráfica:

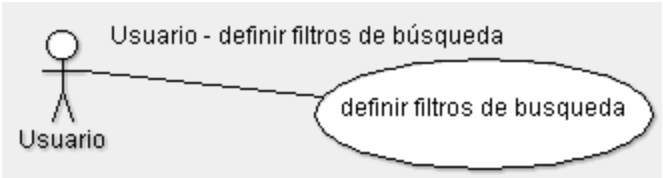

Figura 6 – Caso de uso “definir filtros de búsqueda”

Frecuencia:

Cada vez que un usuario seleccione crear un itinerario turístico.

4.2.1.3. Caso de uso "Seleccionar monumentos".

Descripción:

Caso de uso de nivel medio, a través del cual el usuario seleccionará uno o más monumentos existentes en el módulo de puntos de interés de acuerdo a los filtros de búsqueda anteriormente definidos.

Nivel:

Nivel medio.

Actor principal:

- Usuario.

Actores secundarios:

- Módulo de monumentos.

Prerrequisitos:

- El usuario ha definido los filtros de búsqueda para seleccionar puntos de interés turístico.
- Existen monumentos disponibles a través del módulo de monumentos.

Post-condiciones:

En caso de éxito:

- El sistema conocerá los monumentos que el usuario desea visitar.

En caso de fracaso:

- El sistema desconocerá qué monumentos el usuario desea visitar y por tanto la creación del itinerario se verá interrumpida.
- En caso de no existir monumentos disponibles, se mostrará un mensaje al usuario advirtiendo de este hecho.
- En caso de que el usuario no elija al menos un monumento a visitar, de mostrará un mensaje de error advirtiendo que es necesario elegir al menos un monumento para poder crear un itinerario.

Disparador:

El caso de uso "definir filtros de búsqueda" se ha ejecutado exitosamente.

Escenario principal en caso de éxito:

1. El usuario presiona el botón de búsqueda de monumentos.
2. El sistema recupera la lista de posibles destinos a través del actor "módulo de monumentos".

3. El usuario selecciona uno o más de los monumentos presentados.

Representación gráfica:

Figura 7 – Caso de uso "Seleccionar monumentos"

Frecuencia:

Cada vez que un usuario seleccione crear un itinerario turístico.

Suposiciones:

- Existen puntos de interés turístico en el módulo de monumentos.

4.2.1.4. Caso de uso "Generar itinerario".

Descripción:

Caso de uso de nivel medio, a través del cual la aplicación generará un itinerario turístico.

Nivel:

Nivel medio.

Actor principal:

- Usuario.

Actores secundarios:

- Módulo planificador.

Prerrequisitos:

- El usuario ha seleccionado uno o más monumentos.

Post-condiciones:

En caso de éxito:

- El usuario dispondrá de un itinerario turístico.

En caso de fracaso:

- El usuario no dispondrá de un itinerario turístico.
- En caso de haberse producido un error se mostrará el correspondiente mensaje de error.

Disparador:

El caso de uso "seleccionar monumentos" se ha ejecutado exitosamente.

Escenario principal en caso de éxito:

1. El planificador genera un itinerario turístico de acuerdo al conjunto de puntos de interés turístico definidos en el caso de uso "seleccionar monumentos" y a la lógica definida en el módulo planificador.

Representación gráfica:

Figura 8 – Caso de uso "generar itinerario"

Frecuencia:

Cada vez que un usuario seleccione crear un itinerario turístico.

4.2.1.5. Caso de uso "Recuperar mapas".

Descripción:

Caso de uso de nivel medio, a través del cual el usuario podrá ver los mapas de los puntos de interés que presenten coordenadas y los mapas para las transiciones entre puntos.

Nivel:

Nivel medio.

Actor principal:

- Usuario.

Actores secundarios:

- Módulo de mapas.
- API de Google Maps.

Prerrequisitos:

- Existe un itinerario con puntos de interés turístico definidos.

Post-condiciones:

En caso de éxito:

- El usuario podrá visualizar en el terminal los mapas de los puntos de interés turístico seleccionados y las transiciones entre los mismos.

En caso de fracaso:

- El usuario no podrá visualizar dichos mapas y sólo podrá acceder a la representación textual de los puntos de interés turístico.

Disparador:

El caso de uso "generar itinerario" se ha ejecutado exitosamente.

Escenario principal en caso de éxito:

1. El modulo planificador crea un itinerario turístico
2. La aplicación emplea los datos de dicho itinerario para realizar sucesivas llamadas al módulo de mapas que le irá devolviendo los mapas para cada punto de interés turístico.

Representación gráfica:

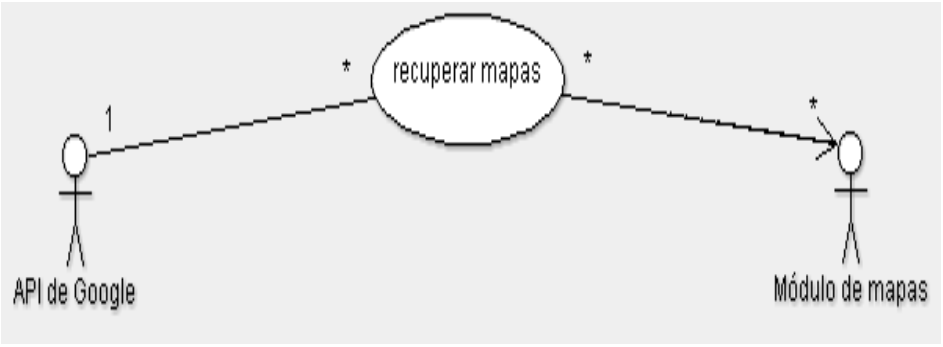

Figura 9 – Caso de uso "recuperar mapas"

Frecuencia:

Cada vez que un usuario seleccione crear un itinerario turístico.

4.2.2. Diagrama de secuencia.

A fin de entender la secuencia lógica a través de la cual un usuario obtiene un itinerario turístico presentamos el siguiente diagrama de secuencia. En él se puede apreciar como primeramente el usuario selecciona una serie de puntos de interés, a continuación el módulo planificador genera un itinerario y finalmente, a la par que el usuario va visionando el itinerario, el módulo de mapas va proporcionando los mapas de los elementos pertenecientes al itinerario seleccionado.

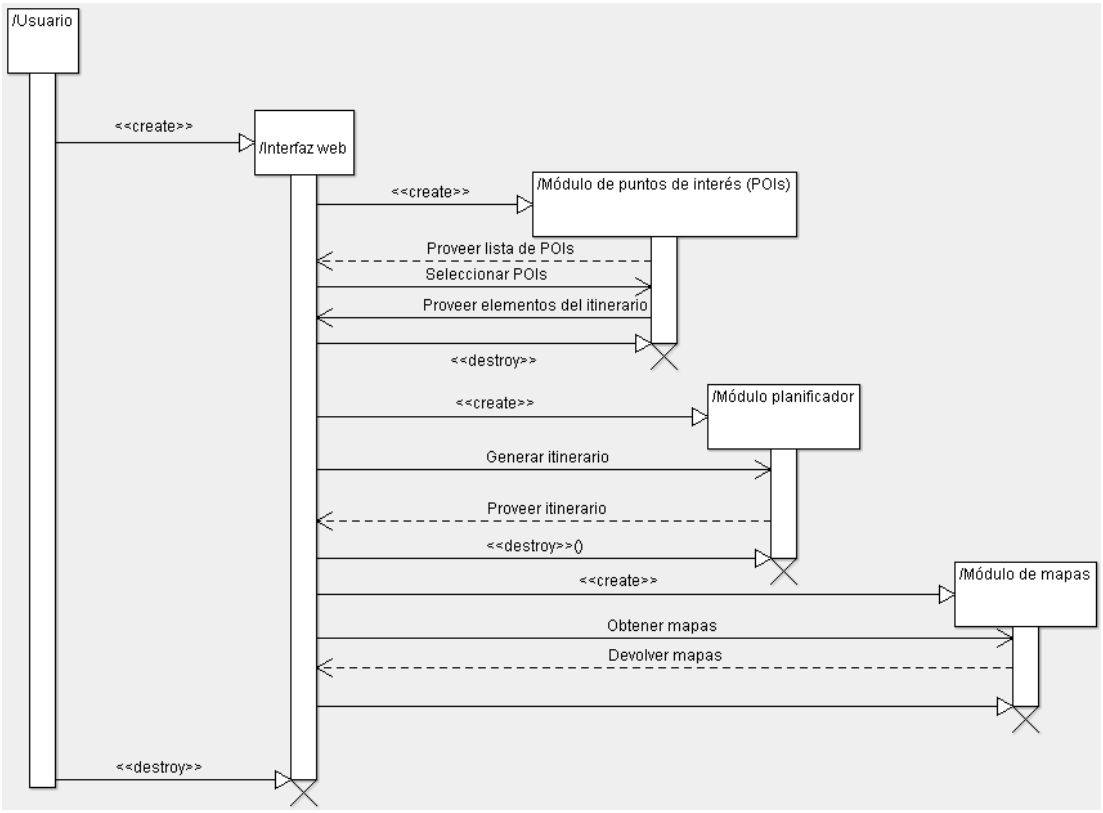

Figura 10 – Diagrama de secuencia

4.2.3. Diagrama de navegación de pantallas.

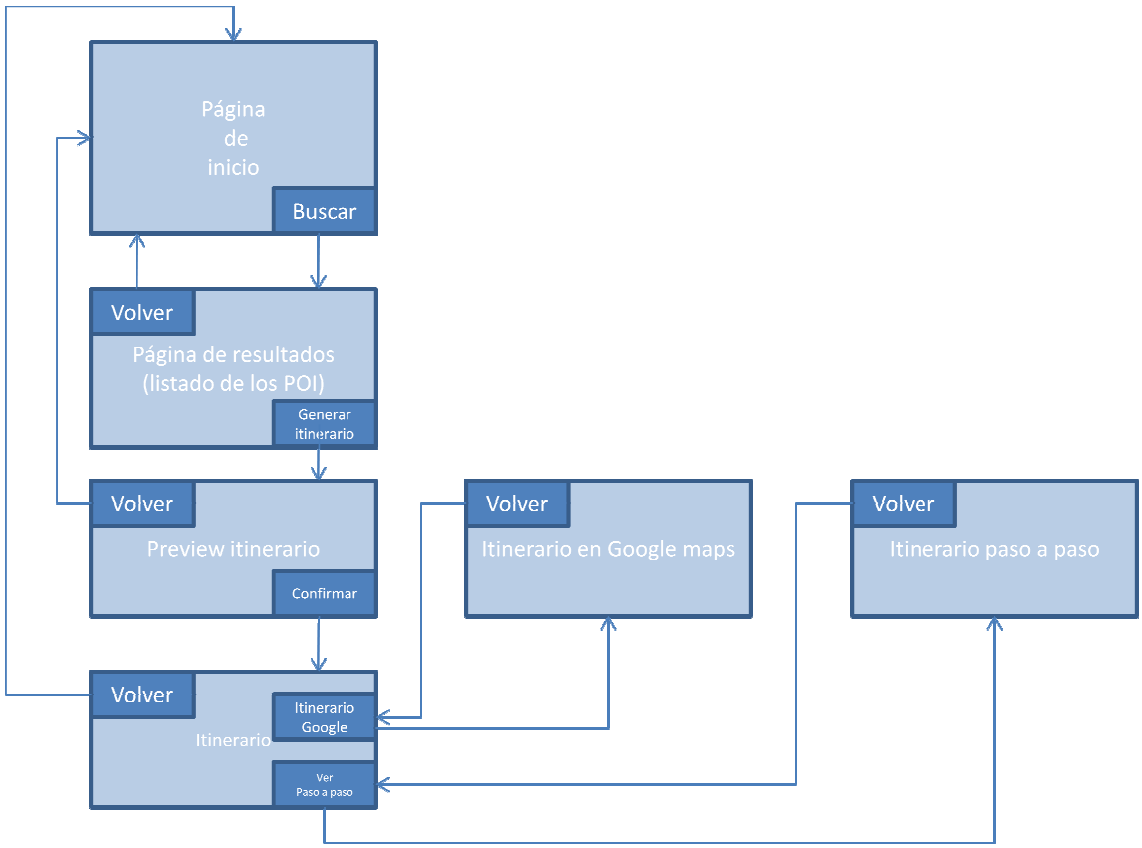

Figura 11 – Diagrama de navegación de pantallas

En el diagrama de navegación de pantallas podemos observar como la navegación se basa en una página principal de búsqueda donde se definen los filtros para la misma. Una vez realizada la búsqueda se nos presenta una lista de resultados los cuales podemos seleccionar para generar el itinerario. Una vez éste ha sido creado podemos ver todos los puntos del mismo o ver el despliegue del mismo paso a paso.

4.3 Estudio del resto de módulos que componen el proyecto.

4.3.1. Módulo de puntos de interés.

El módulo de puntos de interés fue desarrollado por Israel Barroso Pérez como proyecto de fin de carrera en Ingeniería Informática bajo el título "Extracción automática de información turística de Internet". A la hora de realizar el presente proyecto de fin de carrera me fueron proporcionados tanto la memoria como el código fuente de dicho proyecto para su análisis y estudio.

Hablar del "módulo de puntos de interés" es un tanto erróneo dado que en realidad el proyecto comprende dos módulos distintos. El primero de ellos se encarga, a través de un proceso de minería de datos, de acceder a Internet y recopilar información sobre cines, museos y puntos de interés turístico, pudiendo ser estos últimos monumentos, jardines, lugares pintorescos, etc. Dicho proceso analiza las peticiones HTTP realizadas a sitios web de acceso público y a través de una serie de *wrappers* obtiene la información necesaria la cual es almacenada en una ontología. Dado que es un trabajo de investigación, el autor justifica el dominio de los datos analizados a la Comunidad Autónoma de Madrid, considerándose este dominio como suficientemente representativo.

La ontología creada en el módulo anteriormente citado es la base del segundo, el servidor de puntos de interés turístico. Este servidor se encarga de proporcionar una interfaz de comunicación al resto de la aplicación para que los datos almacenados en dicha ontología puedan ser mostrados al usuario. Este segundo módulo fue implementado como una aplicación de Java, la cual ha de ejecutarse cada vez que se quieran recuperar puntos de interés turístico desde la ontología. Al ejecutarse, la aplicación abre un socket y se queda esperando peticiones en formato XML a través de un determinado puerto. Estas peticiones contienen una serie de filtros que delimitan los museos, cines o puntos de interés que son devueltos por el mismo puerto al cliente, igualmente en formato XML. En adelante me referiré a este segundo módulo como la interfaz de búsqueda de museos, cines y puntos de interés, o de un modo más abreviado, interfaz de búsqueda de puntos de interés.

Dado que el objetivo de mi proyecto es comunicarse con esta interfaz para poder consumir la información facilitada por éste módulo, me he centrado principalmente en el análisis de la aplicación Java que facilita el consumo de los datos existentes en la ontología, aceptando cómo válidos los datos existentes en la ontología y no entrando a valorar el proceso de minería de datos que los obtiene.

El proceso seguido para analizar la interfaz de búsqueda de puntos de interés fue el siguiente; dado que disponía del código fuente de la aplicación, instalé la aplicación en mi máquina de desarrollo y ejecuté la aplicación. A continuación implementé un pequeño cliente PHP cuya misión fuese la de enviar una petición XML a la aplicación Java y a la vez capturar el resultado de dicha aplicación. En este momento, comenzaron una serie de dificultades que describo a continuación junto a las soluciones implementadas. Me referiré al cliente implementado en PHP como "el cliente" y al programa Java como "el servidor":

- El servidor espera la petición en el puerto 5000. Este dato no se indica en la memoria del proyecto ni se declara en ningún fichero de configuración, sino que se declara en el código de la aplicación sin más. Tal vez sería más conveniente que este dato se pudiese recuperar de un fichero de configuración, de tal modo que si las características de nuestro entorno no nos permite emplear el puerto 5000 (por ejemplo, por estar ya ocupado), podamos usar un puerto que esté disponible.
- Al lanzar el servidor, se cargan los datos de la ontología. La ontología es un fichero de texto llamado "ontología.txt". La ruta de acceso a dicho archivo estaba definida como una ruta absoluta dentro de la máquina de desarrollo de la persona que desarrollo la aplicación. Dado que yo ejecuté el servidor en mi máquina de desarrollo, la ruta de acceso a la ontología no era correcta, y por este motivo la ejecución finalizaba de modo inesperado al no existir tratamiento de errores referentes a este problema o la posibilidad de configurar dinámicamente la ruta de acceso a la ontología. Este problema fue solucionado implementando una ruta relativa, de tal modo que la localización de la ontología sea independiente de la máquina dónde el servidor se ejecuta. Tal vez en futuras revisiones de este módulo se pueda incrementar el control de excepciones y de situaciones de error a fin de evitar ejecuciones interrumpidas como las que experimentamos durante el proceso de integración.

A continuación describo el proceso de inicio del servidor desde un punto de vista gráfico:

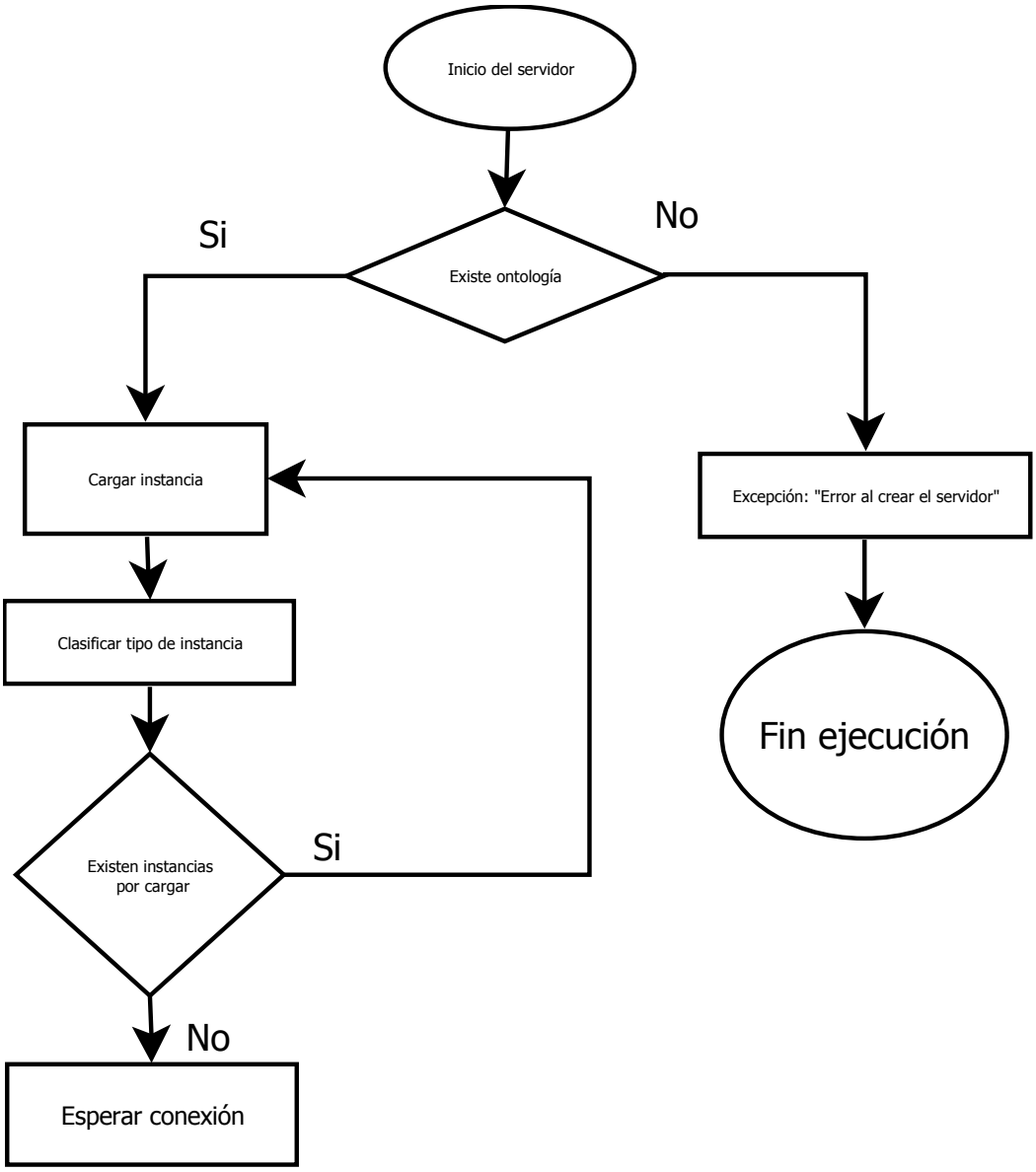

Figura 12 – Diagrama de flujo de inicio del servidor de puntos de interés.

El servidor quedará escuchando la llegada de peticiones en formato XML por el puerto 5000. Una vez que una petición llega, se procede a su análisis y se devuelve la respuesta en formato XML. Veamos la representación gráfica de dicho proceso:

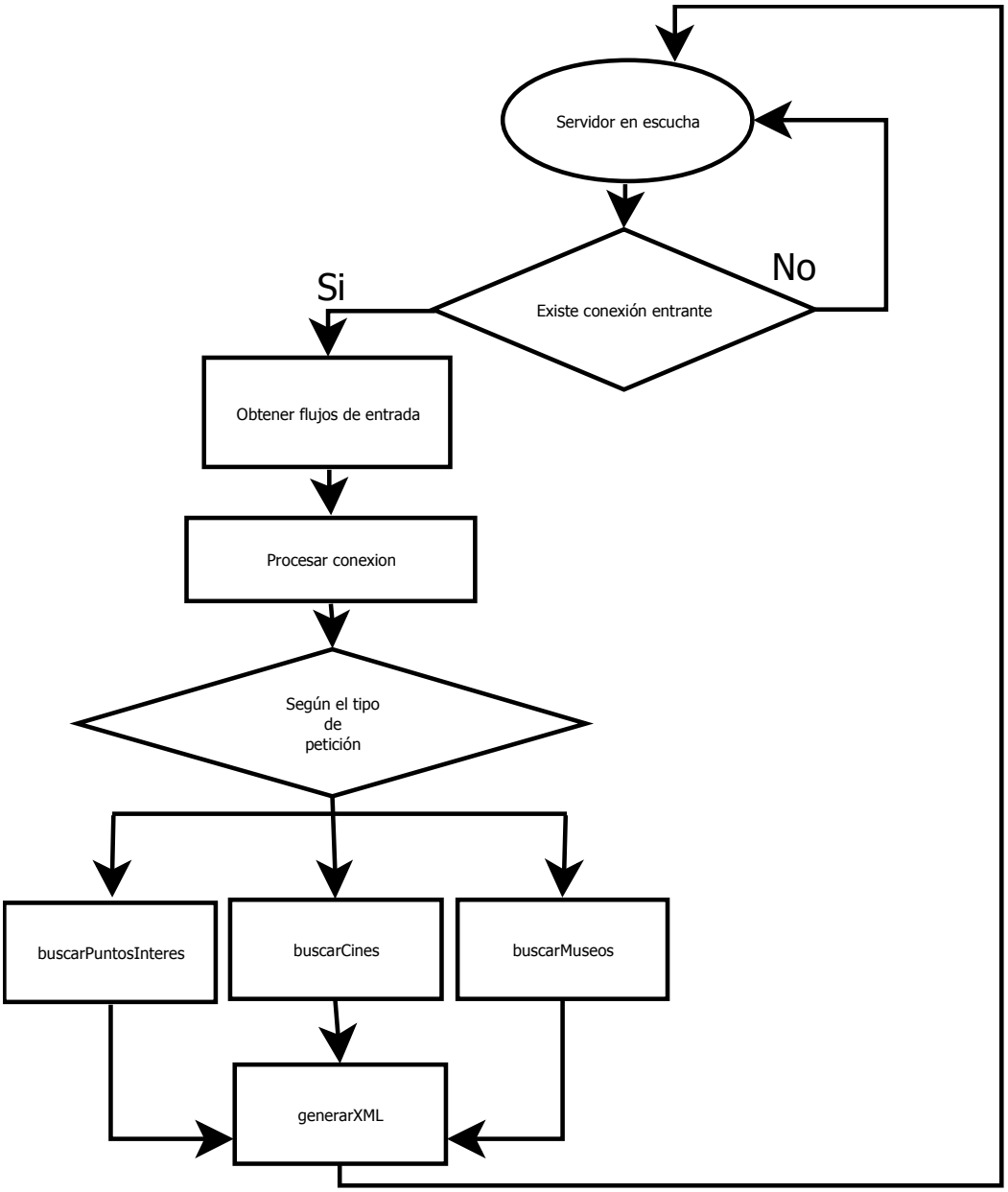

Figura 13 – Diagrama de flujo del procesamientos de una conexión por el servidor de puntos de interés.

Una vez que tenemos nuestro servidor en funcionamiento debería procesar nuestras peticiones, pero al realizar el envío de una petición a través de nuestro cliente PHP observamos que el servidor lanza una excepción y finaliza su ejecución. El motivo de esta excepción es que el servidor espera la petición como un objeto serializado de Java. Esta restricción limita las posibilidades de consumo de esta interfaz de búsqueda a aquellos clientes que sean capaces de implementar objetos serializados de Java. Tal vez

una mejor opción sería la de ofrecer una interfaz más universal que pudiese ser implementada por todo tipo de clientes, por ejemplo un cliente PHP como el que yo he implementado, pero también aplicaciones nativas para iPhone, Android, Symbian, etc las cuales no pueden implementar un objeto serializado de Java. A fin de resolver este problema modifiqué la interfaz de búsqueda de puntos de interés de tal modo que en lugar de esperar los ya mencionados objetos serializados, el servidor esperase peticiones XML en texto plano. El motivo de este cambio es que la implementación original de la interfaz convertía los objetos serializados en cadenas de texto antes de procesar la petición, de este modo se respeta el algoritmo de búsqueda inicial pero se ofrece la posibilidad de que cualquier cliente capaz de enviar peticiones en texto plano puedan consumir la misma. Una pequeña dificultad a la hora de realizar este cambio ha sido el hecho de que la implementación que se hace de "fin de línea – retorno de carro" en Java y PHP es diferente, por tanto el servidor no sabía en qué momento la petición había sido recibida en su totalidad y se quedaba escuchando a la espera de finalizar la recepción. El modo de resolver este problema ha sido empleando un *token* de final de transmisión, de tal modo que el servidor da por concluida la recepción de la petición al aparecer el *token.*

Analizando la arquitectura del servidor existen una serie de consideraciones que nos parece importante enumerar:

- El servidor está diseñado de tal forma que sólo pueda atender a una petición a la vez dado que la respuesta no se envía directamente al cliente, sino que se crea un archivo físico con el contenido de la respuesta y a continuación este archivo se lee para ser enviado al cliente. Ciertamente al tener un solo archivo que es modificado y leído, si el número de clientes que realizan peticiones es elevado, podríamos, al menos teóricamente, devolver una respuesta que no coincide con la petición realizada por el cliente.

- Tal y como hemos descrito anteriormente el servidor realiza búsquedas de cines, museos o puntos de interés, pero no proporciona una interfaz común que pueda combinar resultados de dichas categorías sino que proporciona tres operaciones distintas para cada una de ellas. Esto implica que si queremos realizar una búsqueda combinada de dos o más categorías, deberemos enviar otras tantas peticiones al servidor y gestionar cada una de las respuestas que se producen. Esto hace que la ejecución se vea ralentizada.

• El XML que devuelve la operación para la búsqueda de cines no está correctamente formado dado que una de la etiquetas XML no es correcta, en concreto en lugar de tener la etiqueta de cierre "</RESPUESTAS_CINES>" se tiene la etiqueta "</RESPUESTAS_CINES". He corregido este error tipográfico en el servidor, de tal modo que el XML sea correcto, dado que el mismo producía un error en el proceso de interpretación del archivo XML con las respuestas.

• Una vez se realiza una petición al servidor, el XML resultante de la petición devuelve siempre el conjunto de resultados juntos a los resultados de las peticiones anteriores, es decir, el tamaño del XML crece aritméticamente y muestra resultados erróneos. Esto es debido a que el servidor tiene un *buffer* donde se almacenan aquellos elementos de la ontología que cumplen los criterios de búsqueda. El problema reside en que este *buffer* no se limpia al finalizar el tratamiento de cada petición y por tanto su contenido crece al añadir el resultado de cada petición. He corregido este problema limpiando el *buffer* tras la gestión de cada petición.

Tras analizar el funcionamiento del modulo y su arquitectura, pasaremos a realizar algunas consideraciones acerca de cómo este módulo puede ser mejorado en futuras revisiones:

• La aplicación de servidor de puntos de interés debería estar diseñada para poder responder un elevado volumen de peticiones concurrentes. Con este fin, en lugar de aceptar peticiones únicamente por un puerto, debería ser capaz de abrir puertos de modo dinámico, de tal modo que no existiese un cuello de botella en el puerto 5000. Al mismo tiempo, debería evitarse el hecho de que las respuestas a las peticiones fuesen guardadas en un archivo físico del sistema antes de contestar a la petición. No existe motivo aparente por el cual el resultado de la búsqueda tenga que guardarse. En caso de que así fuese, por ejemplo, para poder realizar informes estadísticos basados en los criterios de búsqueda, sugerimos el empleo de una base de datos a tal fin, pero en principio se podría devolver el resultado de la búsqueda directamente por el socket al cliente.

• Sería óptimo si el servidor ofreciese interfaces de comunicación para cualquier tipo de cliente y no sólo para aquellos que sean capaces de generar objetos serializados de Java. Una idea muy simple es la de construir un servicio web de tipo REST

(*Representational State Transfer*), es decir, un servicio web de tipo ligero al que poder indicar que criterios de búsqueda se desean emplear y que retorne un XML con el conjunto de datos que cumplen dichos criterios.

- Sería muy favorable para la eficiencia del servidor que se pudieran realizar búsquedas que combinasen los distintos tipos de categorías existentes. Puede darse el caso de que un usuario esté interesado en visitar edificios pintorescos, iglesias, museos e ir al cine. Para poder satisfacer las necesidades de este usuario con el servidor actual, el cliente necesitará enviar tres peticiones distintas al servidor, haciendo que la eficiencia de la aplicación disminuya y la navegación se vea afectada por tiempos de espera más largos de lo debido.

4.3.2. Módulo planificador.

El módulo planificador es posiblemente el corazón del proyecto SAMAP en su totalidad. A través de la lógica implementada en este módulo y partiendo de la base de una serie de puntos de interés seleccionados por el usuario en función de sus gustos particulares deberíamos poder generar de modo inteligente un itinerario que se ajustase a esas necesidades.

Dada su complejidad y tamaño, se consideró a este módulo como un proyecto de fin de carrera que debería ser asumido en su totalidad por otro compañero. Lamentablemente a día de hoy, este módulo del proyecto SAMAP no está finalizado y ni tan siquiera en una fase en la que pudiésemos sacar algo de provecho para la integración con el mismo. Por este motivo y tras consultar con la dirección del presente proyecto se tomó la decisión de emular el comportamiento del módulo, aunque haciéndolo de un modo que dejase el camino abierto al compañero o compañera que se encargue de acometer el desarrollo del mismo.

Teniendo en cuenta que el presente proyecto no es el único cliente que podría usar el módulo de planificación, sino que en un futuro tal vez podríamos encontrar otro tipo de implementaciones (por ejemplo, directamente desde aplicaciones de tipo cliente desde el dispositivo móvil), hemos tomado la decisión de definir el interfaz de comunicación con este módulo de la forma más estándar y general posible para favorecer la reusabilidad en el futuro. De este modo tan sólo el núcleo de la planificación tendrá que ser desarrollada.

Por tanto el modelo que proponemos es un servicio web al cuál se le envíen toda la información que el sistema tenga acerca de los puntos de interés turístico que el usuario ha seleccionado. Esto lo consideramos necesario para que el planificador tenga cuanta más información mejor a la hora de realizar su trabajo. El modo de enviar esa petición es en formato XML. El servicio web, al recibir una petición en concreto llevará a cabo tres acciones:

- Primero se guardará el XML enviado dentro del sistema de ficheros del servidor web donde se encuentre el servicio.
- A continuación se ejecutará el planificador, el cual, basándose en el fichero XML creado y en las reglas definidas para el planificador, creará el itinerario turístico.
- Este itinerario turístico en formato XML se devuelve al cliente que solicitaba la planificación.

En nuestro caso concreto, dado que no existen reglas definidas para el planificador hemos ordenado los puntos de interés de modo aleatorio. Dado que el entorno de desarrollo de nuestra aplicación se ha basado en el lenguaje de programación PHP, hemos construido el servicio web que emula la planificación en este lenguaje, no obstante dicho servicio puede ser implementado en cualquier lenguaje de programación que pueda recoger la petición POST en formato XML y devolver el resultado de la planificación en el mismo formato.

Es necesario destacar también, que existe otro módulo a tener en cuenta en este escenario, el módulo de transportes. Este módulo, que sí está definido e implementado, proporciona información sobre los transportes de la zona de interés (en este caso la Comunidad de Madrid), de tal modo que el módulo planificador pueda optimizar el itinerario turístico por ejemplo al calcular los tiempos de espera o al encontrar los medios de transporte que se adecuen a las preferencias del usuario.

El resultado del servicio web será enviado al cliente en el mismo formato XML en el que la petición fue realizada con la salvedad de que los puntos de interés turístico del mismo deberán estar ordenados de acuerdo a la planificación realizada.

4.3.3. Módulo de mapas.

El módulo de mapas fue desarrollado por Alberto Anta Andrés como proyecto de fin de carrera en Ingeniería Técnica en Informática de Gestión bajo el título “Implementación de un módulo de gestión de mapas utilizando herramientas de software libre”. A la hora de realizar el presente proyecto de fin de carrera me fueron

proporcionados tanto la memoria como el código fuente de dicho proyecto para su análisis y estudio. Así mismo tuve ocasión de conocer a Alberto Anta y comentar con él algunas dudas que me surgieron al llevar a cabo la integración de su módulo con mi aplicación.

El objetivo del módulo de mapas es la construcción de una interfaz por la cual el programa principal, en este caso mi aplicación, pueda descargarse de un modo eficiente mapas al sistema de ficheros del servidor para su posterior uso.

Este módulo parte de la gran implantación que existe en el mercado de los SIG (Sistemas de Información Geográfica), muchos de ellos facilitan información de tipo geográfico a terceras partes bien sea de modo gratuito o de modo comercial.

De entre los principales actores existentes en el mercado, Alberto Anta seleccionó Yahoo como el proveedor de servicios de información geográfica y mapas para su proyecto, dado que en su versión no comercial presentaba mejores características que sus competidores, de entre los que destaca Google Maps.

Uno de los objetivos del proyecto de creación de un módulo de mapas era el hacerlo de un modo eficiente. Para esto Alberto diseño un sistema de caché basado en ficheros de Berkeley. Los ficheros de Berkeley son un motor de bases de datos empotrado dado que no lanzan un servicio propio como por ejemplo MySql sino que están empotrados en el propio sistema operativo). Este tipo de base de datos se suelen emplear cuando el sistema necesita un rendimiento muy alto y la naturaleza de los datos permite una indexación basada en claves. La implementación realizada en el módulo de mapas implica que cuando un recurso externo es capturado, por ejemplo un mapa o las coordenadas de una dirección, estos datos son indexados en la caché de tal modo que la próxima vez que sea necesario recuperar estos datos, no haya que “salir” a recuperarlos desde el proveedor, sino que puedan ser recuperados desde la propia caché del módulo, lo que implica que el ancho de banda consumido es mínimo y la velocidad de acceso a los datos es elevada dado que se encuentran en un fichero dentro del propio servidor.

Una de las partes más trascendentales del módulo de mapas es cómo este puede ser integrado por otra aplicación, como es el caso. Para ello el módulo de mapas presenta dos interfaces, una basada en sockets y la otra basada en un servicio web. En cualquiera de los dos casos todas las comunicaciones se llevan a cabo en formato XML.

A través de estas interfaces el módulo presenta una serie de servicios que pueden ser consumidos. Estos servicios presentan el siguiente flujo, comprobación de los parámetros de entrada y de la integridad del XML, verificación de si la misma consulta

ha sido realizada con anterioridad y retorno de información desde la caché en caso afirmativo siempre que no haya caducado, en caso contrario, acceso al servicio web de Yahoo para descargar la información solicitada, una vez descargada la información, se guarda en la caché y se devuelve en formato XML.

Los servicios facilitados por el módulo de mapas son los siguientes:

• Geolocalizador: Servicio cuyo fin es llevar a cabo la geolocalización de una dirección. La geolocalización es el proceso por el cual a través de una dirección postal podemos conocer las coordenadas de dicha localización. Es un paso previo a la captura del mapa en el caso de que las coordenadas del punto no sean conocidas. Dentro del proceso de geolocalización el módulo comprueba que el resultado de la geolocalización no esté presente en la caché y en caso afirmativo, se comprueba que su caducidad no haya expirado. Gráficamente podemos describir este servicio de la siguiente manera:

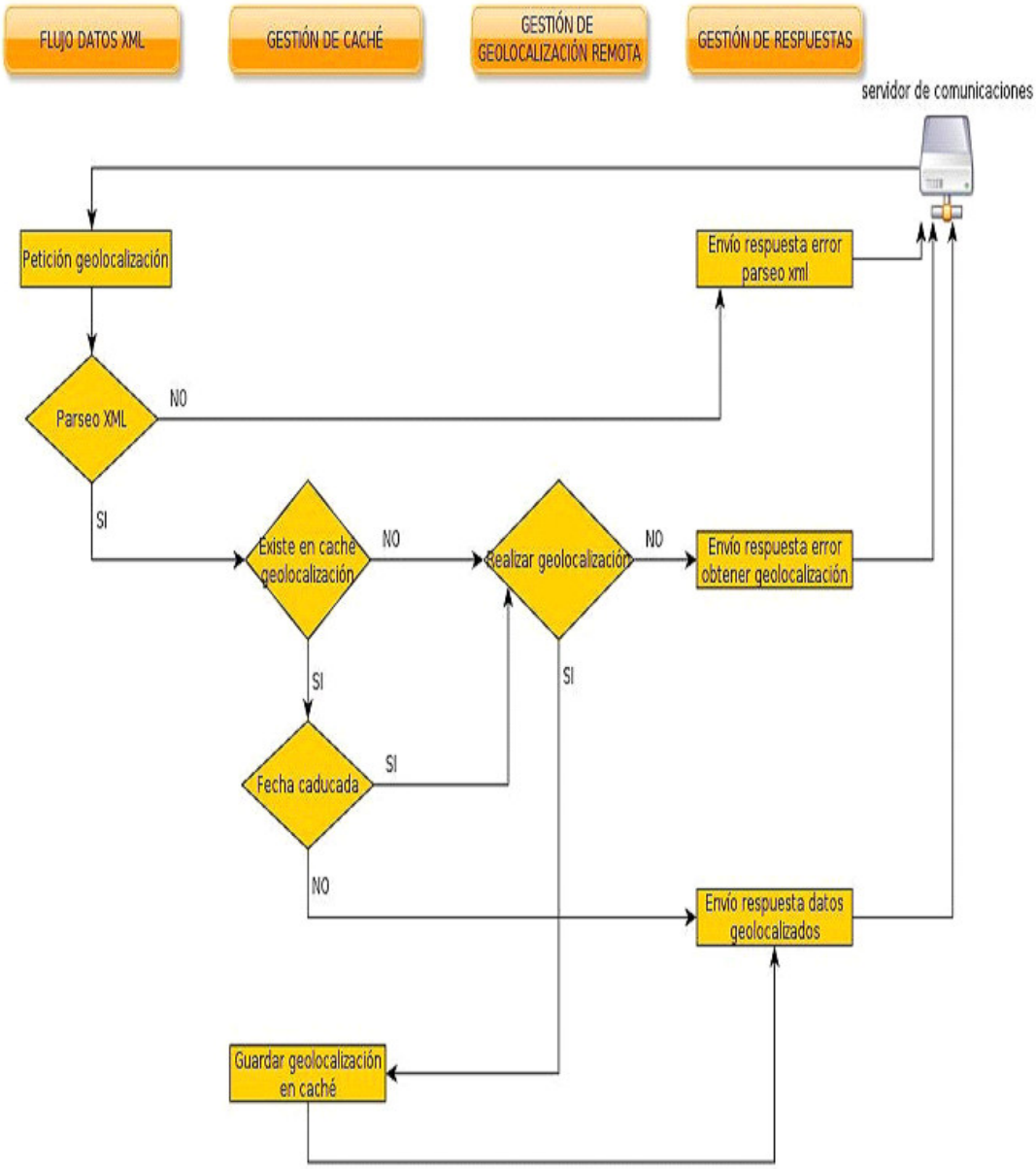

Figura 14 – Diagrama de flujo del proceso de localización geográfica.

• Mapas: Basándose en las coordenadas de un punto, el módulo presenta un servicio que captura el mapa desde los servidores del proveedor de mapas (en este caso Yahoo) y devuelve una url para poder visualizar el mapa capturado. Es importante resaltar que la operación no devuelve la imagen en sí, sino que la guarda en el servidor, para que a través de una dirección web pueda ser recuperada. El diagrama de flujo de esta operación es el siguiente:

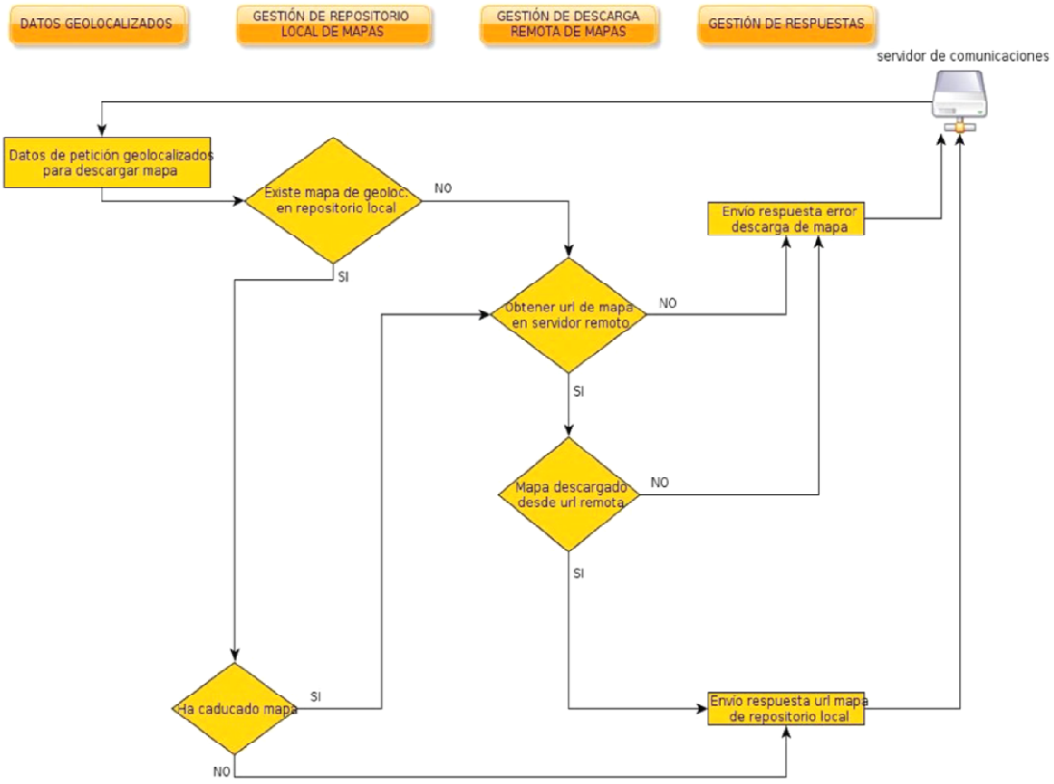

Figura 15 – Diagrama de flujo del proceso de recuperación de mapas.

• Mgm: El servicio MGM es la combinación de los dos anteriores, es decir a partir de una dirección postal se geolocalizan las coordenadas de dicha dirección, se descarga el mapa a un directorio del servidor se y facilita una dirección web para que la imagen pueda ser visualizada.

Adicionalmente el módulo de mapas presenta un servicio web para poder consumir aquellos mapas que están almacenados de modo local en el servidor de mapas. En este caso se realiza una petición a este servicio web y el resultado es la imagen propiamente dicha.

Una vez descrito el funcionamiento del módulo de mapas procederé a comentar la integración realizada de dicho módulo con el presente proyecto.

Primeramente es necesario explicar que dado que el módulo de mapas dispone de dos interfaces para su uso, una basada en sockets y la otra basada en servicios web, he decido emplear esta última por dos motivos. El primer motivo es que la técnica de comunicación de sockets ya se empleó en la integración del módulo de puntos de interés y por tanto he considerado de mayor interés pedagógico el empleo de una técnica diferente en este caso. El segundo motivo es el amplio uso que dentro del mercado se hace del consumo de servicios web basados en comunicaciones XML. Este tipo de interfaces nos permiten tener una vía de comunicación sencilla, pero al mismo tiempo perfectamente estandarizada.

Al haber decidido emplear esta interfaz hemos de realizar dos instalaciones; por un lado los servicios web que nos permiten consumir los servicios ofrecidos por el módulo y por otro lado los archivos que componen el módulo en sí y que proporcionan la funcionalidad anteriormente descrita.

Dado que el presente proyecto ha sido desarrollado en una única máquina o servidor, y ante el hecho de ser la nuestra una aplicación web, nos pareció apropiado instalar los servicios web del módulo de mapas en el mismo servidor web que ya estamos empleando para ejecutar nuestra aplicación. Un pequeño detalle que fue necesario corregir fue el hecho de que la dirección web facilitada por el servicio de mapas (aquél que devuelve la url para recuperar la imagen del mapa) no coincidía con el root de nuestro servidor web. Para solucionar este pequeño contratiempo hemos definido enlaces blandos desde el root del servidor web al directorio donde los archivos residen físicamente, de tal modo que los enlaces puedan emplearse como describe la documentación del módulo de mapas.

En cuanto a la instalación del módulo de mapas propiamente dicho, este se facilita como una serie de archivos organizados en una estructura de directorios. Esta estructura de directorios se ha copiado en el root del servidor UNIX para poder ser invocados por los servicios web anteriormente descritos.

La integración se realizó de un modo satisfactorio y tal vez el único comentario que puede realizarse es que el formato de la información proveniente del servidor de puntos de interés no coincide con los parámetros de entrada de los servicios de mapas en su totalidad. Por ejemplo, el servidor de puntos de interés proporciona un campo llamado dirección que puede tener el valor “Calle Mayor, nº4, 2ºA”, mientras que el

servicio de mapas espera un campo llamado dirección, y un campo distinto llamado número. Al mismo tiempo he detectado dos problemas relacionados con los datos devueltos por el servidor de puntos de interés.

El primero es la gran presencia de abreviaturas en el conjunto de datos devueltos por el servidor de puntos de interés, los cuales, al ser enviados al servidor de mapas no devuelven un resultado satisfactorio. Una solución simple a este problema es la de llevar a cabo un mapeo de dichas abreviaturas antes de consumir el servicio de mapas. Por ejemplo, podríamos mapear "Av." con "Avenida", "C.C." con "Centro Comercial", "Km." con "Kilómetro" etc.

El segundo problema es la carencia de coordenadas por parte de muchos de los puntos de interés turísticos proporcionados por el servidor de puntos de interés. Por este motivo no sólo el servicio de captura de mapas ha sido usado, sino que el servicio de geolocalización también ha sido necesario y pese a ello bastantes localizaciones no se han podido encontrar. Esto es debido a que muchos puntos de interés no tienen una calle y un número como dirección sino que al estar situados fuera del centro urbano o en el medio de una carretera, los servicios de Yahoo no los pueden geolocalizar. Este fenómeno está especialmente presente en el caso de los cines, ya que muchos de ellos están ubicados en centros comerciales que carecen de una dirección exacta.

Es necesario comentar que los servicios de geolocalización de Yahoo no son ni mucho menos infalibles y si por ejemplo carecemos de la información sobre el municipio y únicamente indicamos el nombre de la calle, puede que el resultado nos ubique en un lugar erróneo.

4.4 Interfaces de comunicación con el resto de módulos.

4.4.1. Comunicación con el módulo de puntos de interés.

Tal y como he definido en el apartado de estudio del resto de los módulos que componen el proyecto, el servidor de búsqueda de puntos de interés presentaba ciertos errores o carencias que hacían complicado su uso por parte de un cliente. Una vez que estos problemas han sido solventados procedo a describir como ha sido diseñada la interfaz de comunicación con dicho módulo.

En primer lugar y dado que el objetivo de esta integración es la de poder realizar búsquedas desde la aplicación web, la cual está basada en un CMS, consideramos que una buena decisión sería la de llevar a cabo la comunicación con el servidor a través de un módulo de dicho CMS que se encargase de todo aquello referido

a la misma. Dado que el CMS se desarrollará en PHP, lo ideal es poder comunicarnos con el servidor a través de PHP.

Afortunadamente PHP posee de forma nativa funciones para implementar sockets de comunicación, especialmente dos de ellas: *socket_create* y *socket_connect*, las cuales nos permitirán comunicarnos con el socket que nos ofrece el servidor.

Ciertamente el usuario habrá de definir los criterios de búsqueda que han de ser enviados al servidor y será trabajo de este módulo, el procesar dichos criterios, comprobar que son correctos, construir un XML válido a partir de ellos, implementar aquellos pasos necesarios para enviarlo a través del socket al servidor, recibir la respuesta y procesarla de modo que el CMS pueda trabajar con ella.

Tal y como definimos con anterioridad, el servidor de búsqueda de puntos de interés ofrece tres operaciones básicas (buscar puntos de interés, buscar cines y buscar museos), las cuales no presentan los mismos criterios de filtrado. Por ese motivo pasamos a presentarlos de modo específico para cada operación:

- Búsqueda de puntos de interés (PETICION_PI):

Nombre	Descripción	Ejemplo
Nombre	Nombre del punto de interés	Academia de Cine
Direccion	Dirección	C/. Zurbano 3
Localidad	Localidad	Madrid
Codigo_Postal	Código Postal	28010
Telefono	Número de teléfono	916613909
Metro	Parada de metro más cercana	Línea 5: Callao
Autobus	Líneas de autobús cercanas	3, 60 y 148
Descripcion	Descripción del lugar	n/a*
Horario	Horario de apertura	n/a*
Latitud	Latitud geográfica	40.419451
Longitud	Longitud geográfica	3.6917179
Tipo	Tipo de punto de interés	Edificio civil

Tabla 1 – Conjunto de criterios de filtrado para la operación PETICIÓN PI.

* No existen datos en la ontología facilitada

- Búsqueda de museos (PETICION_MUSEOS):

Nombre	Descripción	Ejemplo
Nombre	Nombre del museo	Alcion Art Gallery
Direccion	Dirección	C/. Orellana 14
Localidad	Localidad	Madrid
Codigo_Postal	Código Postal	28004
Telefono	Número de teléfono	91 319 30 37
Zona	Zona de la ciudad	CENTRO
Fax	Número de fax	91 319 30 37
VentaEntradas	Precio o tipo de entrada	entrada gratuita
Horario	Horario de apertura	De 10 a 18 horas
Web	Página web	http://www.bne.es
Latitud	Latitud geográfica	40.419451
Longitud	Longitud geográfica	3.6917179

Tabla 2 – Conjunto de criterios de filtrado para la operación PETICIÓN_MUSEOS.

- Búsqueda de cines (PETICION_CINES):

Nombre	Descripción	Ejemplo
Nombre	Nombre del cine	Acteón
Direccion	Dirección	Pza. Carmen, 7
Telefono	Número de teléfono	915 222 281
Email	Dirección de email	manoteras@ugc.es
Precio	Precio de la entrada	6,80 €
Precio_Reducido	¿Existe descuento? Precio	J y matinal 5,50 €
Municipio	Municipio	Madrid
Zona	Zona de la ciudad	Centro

Tabla 3 – Conjunto de criterios de filtrado para la operación PETICIÓN_CINES.

Por tanto, el componente a implementar necesitará gestionar cada una de las peticiones de modo independiente. Ofreceré más detalles sobre cómo el componente ha sido implementado en el capítulo referido a la implementación de la aplicación.

4.4.2. Comunicación con el módulo planificador.

Tal y como definimos en el apartado anterior el módulo de planificación nos ofrecerá un servicio web al cual poder enviar una petición en formato XML con los puntos de interés a partir de los cuales se deberá planificar el itinerario turístico.

El problema de esta comunicación estriba en el hecho de que los puntos de interés no son uniformes, es decir, dependiendo de si estamos hablando de un museo, un cine o un edificio religioso, el sistema conoce una serie de atributos u otros. Ante esta realidad no podemos definir un estándar estático para los campos que estarán presentes en la petición XML, dado que estos dependerán del tipo de punto de interés que estemos tratando. Otra opción sería la de definir el subconjunto de los atributos comunes a todos los tipos de puntos de interés y sólo enviar estos atributos en la petición, pero de este modo perderíamos información valiosa para la planificación como por ejemplo los horarios de las sesiones de los cines o de apertura de los museos.

La dirección del servicio web donde reside el planificador puede ser configurada a través del archivo de configuración del CMS. En nuestro caso la hemos definido en la siguiente dirección:

http://localhost/planner/webserver.php

El formato genérico del XML a enviar será:

```
<?xml version=”1.0” encoding=”utf-8” ?>
  <items>
    <item>..</item>
    <item>..</item>
    <item>..</item>
     ..
   <item>..</item>
  <items>
```

El formato de los “ítem” se describe a continuación para cada uno de los tipos de puntos de interés:

Item de tipo “Museo”

```
<item>
   <Nombre>Alcion Art Gallery</Nombre>
   <Direccion>C/. Orellana 14</Direccion>
   <Localidad>MADRID</Localidad>
   <Codigo_Postal>28004</Codigo_Postal>
   <Telefono>91 319 30 37</Telefono>
   <Zona>CENTRO</Zona>
   <Fax>91 319 30 37</Fax>
   <VentaEntradas>entrada gratuita</VentaEntradas>
   <Horario>L a V: 10.30 a 13.30 y 17.00 a 20.30</Horario>
   <Web />
   <Longitud>-3.6944590000000002</Longitud>
   <Latitud>40.426166000000002</Latitud>
   <Tipo>Museo</Tipo>
</item>
```

Item de tipo “Punto de Interés”:

```
<item>
   <Nombre>Campo del Moro</Nombre>
   <Direccion>Gta. San Vicente s/n</Direccion>
   <Localidad>MADRID</Localidad>
   <Codigo_Postal>28008</Codigo_Postal>
   <Telefono/>
   <Metro>Linea 6 y 10: Principe Pio - RENFE: Principe Pio</Metro>
   <Autobus>25, 33, 39, 41, 46, 68, 69, 75, 138 y C</Autobus>
   <Descripcion/>
   <Horario/>
   <Tipo>Jardines y zonas verdes</Tipo>
   <Longitud>-3.7204790000000001</Longitud>
   <Latitud>40.419936</Latitud>
</item>
```

Item de tipo “Cine”:

```
<item>
   <Nombre>Acteon</Nombre>
   <Direccion>Pza. Carmen, 7</Direccion>
   <Telefono>915 222 281</Telefono>
   <Email />
   <Precio>7 €</Precio>
   <PrecioReducido>X y primera sesion</PrecioReducido>
   <Municipio>Madrid</Municipio>
   <Zona>Centro</Zona>
   <Tipo>Cine</Tipo>
</item>
```

Tal y como comentamos con anterioridad el resultado del servicio web será un XML con el mismo formato que la petición, con la salvedad de que los ítems presentes en el mismo deberán estar ordenados de acuerdo a la planificación determinada por el módulo planificador.

4.4.3. Comunicación con el módulo de mapas.

Con anterioridad comentamos que de las dos posibilidades que el módulo de mapas ofrece usaremos la que está basada en servicios web en lugar de la versión basada en sockets. Estos servicios web son totalmente independientes y cada uno tiene su propio script de acceso, el formato de los mismos aparece perfectamente documentados en la documentación de dicho módulo. A continuación pasamos a describir la interfaz de las tres operaciones anteriormente mencionadas para el módulo de mapas.

Servicio de geolocalización - descripción del servicio:

Servicio:	Geolocalización
Objetivo:	Geolocalización de la dirección indicada y almacenamiento en caché si procede
Url:	http://localhost/webservice/geolocalizador.html

Tabla 4 – Descripción del servicio de geolocalización.

Servicio de geolocalización - parámetros del servicio:

Nombre:	**Descripción**	**Ejemplo**
direccion:	nombre de la dirección a geolocalizar	Avenida de la Universidad
numero:	número de la dirección anterior	1
Ciudad	Nombre de la ciudad	Leganés
Cp	código postal de la dirección	28911
Provincia	nombre de la provincia	Madrid
localización	Todos los campos anteriores agrupados en un solo parámetro.	Avenida de la Universidad, 1, Leganes, 28911, Madrid

Tabla 5 – Parámetros del servicio de geolocalización.

Ejemplo de llamada al servicio:

http://localhost/webservice/geolocalizador.html?direccion=calle+barrionuevo&numero=5&ciudad=leganes&cp=28911&provincia=madrid&localizacion=calle+barrionuevo+5+leganes+28911+madrid

Respuesta en caso de éxito:

```
<?xml version=”1.0” encoding=”utf-8” ?>
  <respuesta>
    <error>0</error>
    <latitud>-3.768355</latitud>
    <longitud>40.328999</longitud>
  </respuesta>
```

Servicio de descarga de mapas - descripción del servicio:

Servicio:	Descarga de mapas
Objetivo:	Descarga del mapa e información de la ubicación del mapa y almacenamiento en caché si procede.
Url:	http://localhost/webservice/mapas.html

Tabla 6 – Descripción del servicio de geolocalización.

Servicio de mapas - parámetros del servicio:

Nombre:	**Descripción**	**Ejemplo**
Longitud	Longitud del punto a localizar en el mapa	40.328999
Latitud	Latitud del punto a localizar en el mapa	-3.768355

Tabla 7 – Descripción del servicio de descarga de mapas.

Ejemplo de llamada al servicio:

http://localhost/webservice/mapas.html?longitud=-2.66860&latitud=40.84942

Respuesta en caso de éxito:

```
<?xml version=”1.0” encoding=”utf-8” ?>
    <respuesta>
        <error>0</error>
        <url_mapa>http://localhost/maps/27/1_40.84942_-
2.668650_500x620.gif</url_mapa>
    </respuesta>
```

Servicio global MGM - descripción del servicio:

Servicio:	MGM (Módulo Gestor de Mapas)
Objetivo:	Geolocalización de la dirección indicada y descarga del mapa e información de la ubicación del mapa y almacenamiento en caché de ambas si procede.
Url:	http://localhost/webservice/mgm.html

Tabla 8 – Descripción del servicio global MGM.

Servicio global MGM - parámetros del servicio:

Nombre:	**Descripción**	**Ejemplo**
direccion:	nombre de la dirección a geolocalizar	Avenida de la Universidad
numero:	número de la dirección anterior	1
Ciudad	Nombre de la ciudad	Leganés
Cp	código postal de la dirección	28911
Provincia	nombre de la provincia	Madrid
localización	Todos los campos anteriores agrupados en un solo parámetro.	Avenida de la Universidad, 1, Leganes, 28911, Madrid
Latitud	Latitud del punto a localizar en el mapa	40.328999 [OPCIONAL]
Longitud	Longitud del punto a localizar en el mapa	-3.768355 [OPCIONAL]

Tabla 9 – Parámetros del servicio global MGM.

Ejemplo de llamada al servicio:

http://localhost/webservice/mgm.html?direccion=calle+barrionuevo&numero=5&ciudad=leganes&cp=28911&provincia=madrid&localizacion=calle+barrionuevo+5+leganes+28911+madrid

Respuesta en caso de éxito:

```
<?xml version=”1.0” encoding=”utf-8” ?>
```

```
<respuesta>
    <error>0</error>
    <latitud>-3.768355</latitud>
    <longitud>40.328999</longitud>
    <url_mapa>http://localhost/maps/27/1_40.84942_-
2.668650_500x620.gif</url_mapa>
</respuesta>
```

Una vez definidas las interfaces que se han usado simplemente definir el modo de hacerlo. Ciertamente se podrían haber usado directamente desde las páginas del CMS, pero dada la relación existente entre los servicios del módulo, las he juntado dentro de un componente del CMS. Un componente del CMS no es más que una clase de PHP que aglutina funciones y atributos que de alguna manera están relacionados lógicamente. Cada vez que la aplicación necesite hacer uso del módulo de mapas, no habrá más que instanciar esta clase y consumir sus operaciones.

5. DISEÑO E IMPLEMENTACIÓN DEL SISTEMA

5.1 Diseño del módulo gestor de comunicaciones.

Uno de los objetivos principales del presente proyecto es la implementación de un módulo gestor de comunicaciones entre la aplicación que el usuario empleará y el resto de los módulos que componen el proyecto SAMAP.

En los apartados anteriores hemos presentado dichos módulos y las interfaces de comunicación que presentan. Al mismo tiempo hemos definido el tipo de solución a implementar, que estará basada en un gestor de contenido para crear una aplicación web programado en PHP.

A la hora de diseñar el módulo gestor de comunicaciones y tras haber analizado los módulos anteriores constatamos que cada uno de los módulos presenta interfaces diferentes y por tanto es complicado implementar un único módulo que sea lo suficientemente flexible como para comunicarse con el resto. En su lugar lo que haremos será implementar la lógica perteneciente a este módulo en componentes independientes del gestor de contenidos, de tal modo que sea el gestor de contenidos el que lleve el peso de la lógica de negocio mientras que los componentes se comportaran como *plug-ins* que se emplearan cuando sea necesario realizar una comunicación concreta. Con este diseño conseguimos tener componentes más simples, con una funcionalidad más especializada y que sólo se instancia cuando sea estrictamente necesario. El diseño concreto de cada componente se presentará a continuación cuando pasemos a desarrollar los componentes del CMS.

5.2 Diseño de la aplicación web.

Tal y como presentamos anteriormente la solución diseñada para la creación de la aplicación web se basa en un gestor de contenido que producirá las páginas XHTML deseadas. El motivo de haber seleccionado esta solución se basa en los siguientes conceptos:

- Principio arquitectónico de modelo vista controlador (MVC): Este principio separa los datos, la interfaz de usuario y la lógica de control en tres componentes independientes.
- Posibilidad de comunicarse a través de diferentes componentes y/o interfaces con el resto de módulos del sistema desarrollados por otros desarrolladores.

• Posibilidad de reutilizar la funcionalidad presente en los componentes en cualquiera de las páginas de la aplicación.

• Posibilidad de adaptar el componente gráfico a las características del terminal desde el que se está consumiendo la aplicación.

Tal y como se presenta en el siguiente gráfico el CMS presenta una estructura de cinco partes claramente diferenciadas en función de su funcionalidad dentro del proceso de creación de una página XHTML.

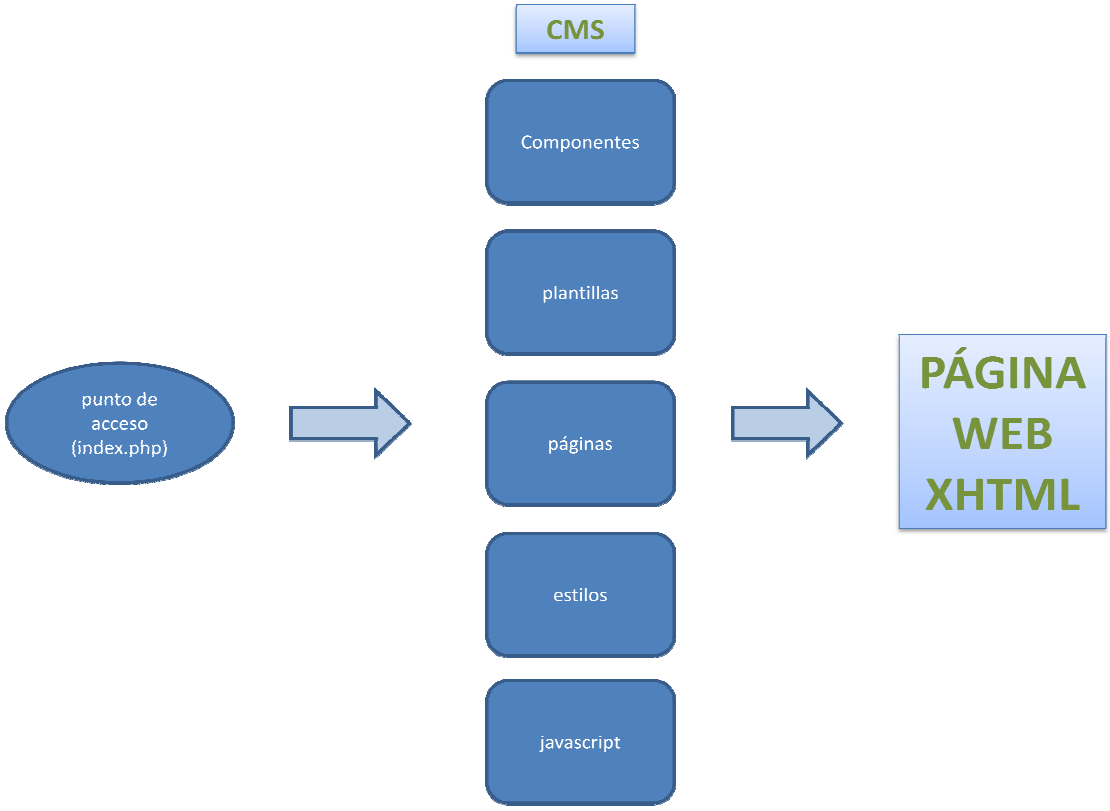

Figura 16 – Estructura del CMS.

Cada una de estas cinco partes se presenta a continuación:

• **Componentes**: El gestor de contenido necesita de funcionalidad, la cual ha de estar correctamente estructurada de acuerdo a la lógica del sistema. Para facilitar el uso de esta funcionalidad hemos definido los componentes como un modo de agrupar las operaciones necesarias de un modo lógico. Existirán componentes para comunicarse con otros módulos, para generar el código XHTML o para consumir recursos externos.

A un bajo nivel son clases de PHP que pueden ser instanciadas en aquellas partes que el gestor de contenidos considere necesarias.

• **Plantillas**: Combinación de elementos de XHTML y marcas de sustitución. El código XHTML nos proporcionará la estructura necesaria para que el navegador sepa mostrar la página web, mientras que las marcas de sustitución serán reemplazadas por la información relevante para el usuario. La existencia de plantillas con marcas de sustitución es muy importante para los diseñadores gráficos, dado que les permite definir la disposición de los elementos visuales de la página sin tener que conocer la lógica existente en el programa.

• **Páginas**: Tradicionalmente una aplicación web se compone de un conjunto de páginas con las que el usuario va interactuando de tal modo que se pueda establecer una comunicación bidireccional con el servidor web. En esta comunicación el usuario introduce información al sistema que la lógica implementada en el servidor web utiliza para definir la información que sucesivamente se va mostrando en la información, es decir, las páginas actúan como gestores de qué información se muestra al usuario y qué se hace con la información que el usuario envía al servidor web. Teniendo en cuenta que hemos definido que usaremos un modelo vista controlador, vemos que serán las páginas las que controlen la lógica del flujo de la navegación dentro de la aplicación web. En cada página se instanciarán los componentes que son necesarios para proveer la funcionalidad a la página (controlador), así como las plantillas asociadas a las mismas (vista).

• **Estilos**: Cuando implementamos una página XHTML tenemos una serie de elementos visuales que presentan la información que deseamos comunicar, pero normalmente nuestro objetivo no es sólo presentar un conjunto de datos al usuario sino que además este conjunto de datos tengan un cierto estilo. En la programación web para proveer de distintos estilos gráficos a una solución se emplean las hojas de estilo, en inglés, *Cascade Style Sheet* o CSS. Las hojas de estilo permiten definir la presentación de los elementos gráficos de la página XHTML generada. Dado que el producto de nuestro CMS va a ser visualizado en un dispositivo móvil este apartado es esencial a fin de que el usuario tenga una experiencia satisfactoria en el uso de la aplicación. Más adelante profundizaremos en las características peculiares de los estilos en los dispositivos móviles.

• **JavaScript**: Hasta ahora hemos hablado de páginas y componentes que serán ejecutados en el lado del servidor, pero cada vez es más frecuente que los dispositivos móviles tengan capacidad de ejecutar scripts en sus navegadores móviles (generalmente javaScript). De este modo el gestor de contenidos puede disponer de ciertas funciones implementadas en javaScript para ser ejecutadas en el propio terminal. De este modo la navegación es más rica para el usuario, dado que no es necesario enviar información al servidor web para ejecutar estas funciones, sino que se ejecutan íntegramente en el cliente.

5.3 Los componentes.

A continuación pasamos a describir los componentes existentes en la aplicación, todos ellos se han implementado como clases de PHP que pueden ser usados por la aplicación web.

5.3.1. Component_cms.php.

El componente "cms" es el principal componente del sistema. A través de él se articula la creación de las páginas XHTML de la aplicación. Todas las páginas de la aplicación web han de instanciar este componente de modo que se pueda enviar la página al dispositivo móvil.

Al instanciar la clase, se carga un fichero de configuración llamado *settings.ini* en él se pueden definir aquellos valores que serán estáticos a lo largo de la aplicación. Estos valores deben estar definidos del modo *clave = "valor"* de tal modo que al ser cargados por el componente queden almacenados en el atributo *settings* del mismo. Una vez cargado el archivo de configuración se cargan todas las plantillas del sistema, de tal modo que cuando el gestor de contenidos solicite reemplazar algunas de las marcas de reemplazamiento por su valor real, dichas plantillas ya se encuentren en memoria.

A continuación describimos algunas de las funciones más importantes del componente:

- *makePage:*

La función *makePage* es la función principal del componente dado que se encarga de componer y producir la página. El objetivo principal de esta función es definir los elementos existentes en la página. Esto se realiza a través de los pasos lógicos que enumeramos a continuación:

– Definición del tipo de documento *(DOCTYPE):*

El DOCTYPE es una etiqueta XHTML que define el tipo de documento con el que estamos trabajando. A la hora de generar páginas XHTML para dispositivos móviles tenemos un DOCTYPE específico para los mismos.

- Definición de las meta etiquetas de la página:

Las meta etiquetas son elementos HTML que carecen de representación visual en el resultado visual de la página, pero son de vital importancia para el navegador web a la hora de conocer metadatos sobre la página, por ejemplo quién es el autor, el título, cuándo fue creada, con qué herramienta, etc.

- Definición de recursos externos:

Tal y como hemos definido anteriormente a la hora de crear una página consumiremos recursos externos tales como librerías de javaScript u hojas de estilo. La localización de dichos recursos ha de definirse dentro de la etiqueta <*head*> del documento XHTML.

- Identificación de la página a cargar:

Durante todo el proceso de navegación, la aplicación ha de conocer en que página se encuentra, dado que en función de la página, unos recursos u otros serán usados. Una vez identificada la página se procederá a su inclusión y se interpretará el código de la misma. El proceso de carga de la página implica el consumo de datos y recursos diversos, pero tiene siempre como objetivo final la definición del contenido del atributo *content* del componente.

- Identificación de las marcas de sustitución, y reemplazamiento de las mismas.

Una vez realizado el paso anterior, el contenido presenta una serie de etiquetas XHTML junto a una serie de marcas de sustitución. El componente tendrá que detectar dichas marcas, reemplazarlas por su valor real y guardar el resultado de la operación en el atributo *content.* Uno de los factores que dan mucha versatilidad a este componente es la posibilidad de incluir recursividad dentro de las marcas de sustitución, de tal modo que una plantilla, identificada por una marca de sustitución puede a su vez incluir nuevas marcas de sustitución que relacionen a otras plantillas diferentes. Este hecho permite una atomización de las plantillas que hace muy simple su mantenimiento.

- Limpieza de las marcas de sustitución no usadas

Una vez realizado el proceso de reemplazamiento de las marcas de sustitución, puede ser que existan algunas marcas que no han sido reemplazadas, es decir, el CMS no ha definido el valor para las mismas. Por este motivo han de ser eliminadas del XHTML final, de tal modo que el resultado de la página quede libre de las mismas. Esta

operación que puede parecer algo enrevesada nos permite dotar de un gran dinamismo al CMS, dado que podremos definir plantillas con diversas marcas de reemplazamiento para que dependiendo de cómo los datos varíen sean unas marcas u otras las que se reemplazan, mientras que el resto simplemente desaparecen.

- impresión del resultado del proceso en forma de página XHTML.

Finalmente el componente muestra el resultado de la página para que pueda ser consumida por el usuario.

- *getContent:*

La función *getContent* permite recuperar el contenido de una plantilla para poder manipularlo. Dentro de esta operación comprobamos si existen marcas de sustitución que ya hayan sido definidas en el atributo *"content"* de tal modo que no exista posibilidad de definir dos veces la misma marca.

- *setContent:*

Función que define el valor de una marca de sustitución con un valor concreto. El proceso se divide en dos pasos. Primeramente se comprueba si la marca está presente en alguno de los valores ya definidos en el atributo *"content"* y finalmente se añade ese valor a dicho atributo, de tal modo que al crear la página, su valor pueda ser usado para futuros reemplazamientos si así fuese necesario.

5.3.2. Component_poi.php.

El componente POI (del inglés *points of interest*) es el encargado de implementar la parte del modulo de gestión de comunicaciones en lo que se refiere al servidor de puntos de interés. Tal y como definimos con anterioridad la comunicación se realiza a través de sockets y tiene como objetivo la búsqueda de puntos de interés de acuerdo a las preferencias del usuario. No obstante tenemos una complicación, dado que el interfaz del servidor de puntos de interés presenta tres operaciones diferentes, en función del tipo de punto de interés sobre el cual se vaya a realizar la búsqueda. En la práctica esto significa que el XML con la petición de búsqueda que enviaremos a través del socket variará en función de si lo que nos interesan son cines, museos o puntos genéricos de interés turístico. A fin de poder llevar a cabo del proceso de búsqueda de la forma más estructurada posible lo que haremos será tener un *parser* específico para cada operación. Este *parser* producirá una petición XML que será la que enviemos a través del socket hacia el servidor. El componente quedará a la espera del resultado de

la petición y una vez que esta se haya completado, transformará el resultado en un array de PHP, más manejable que un XML, a fin de mostrar el resultado de la búsqueda.

Las funciones principales del componente de puntos de interés son:

- *parseGetMuseos:*

Función que lleva a cabo el filtrado de la petición para recuperar museos del servidor de puntos de interés.

- *parseGetPi:*

Función que lleva a cabo el filtrado de la petición para recuperar puntos de interés genéricos del servidor de puntos de interés.

- *parseGetCines*

Función que lleva a cabo el filtrado de la petición para recuperar cines del servidor de puntos de interés.

- *getTcpResponse*

Función que implementa la conexión con el servidor de puntos de interés y define todos los parámetros para la ejecución de la misma. Es esta función en donde se definen el puerto por el cual se va a lanzar el socket, la dirección donde se encuentra el servidor, etc. Además y dado que el cliente y el servidor están implementados con distintos lenguajes de programación es necesario gestionar el final de la comunicación, lo cual se realiza a través de un *token* o conjunto de caracteres, especialmente definidos para la ocasión.

- *returnResult:*

Función que a partir de una petición XML lanza la función getTcpResponse, recoge su resultado y lo procesa para que pueda ser usado por el CMS.

- *simpleXml2Array:*

Función que convierte un XML en un *array* de PHP, dado que el manejo de *arrays* es más simple y eficiente en PHP que el manejo de objetos de tipo XML.

5.3.3. Component_mapas.php.

El componente mapas es el encargado de implementar la parte del modulo de gestión de comunicaciones en lo que se refiere al servidor de mapas. Este componente es simplemente un envoltorio para el consumo de dos servicios web que ofrece el servidor de mapas. Las dos operaciones existentes son las siguientes:

- *getMap:*

Función que a partir de unas coordenadas geográficas realiza una llamada al servidor de mapas para recuperar el mapa desde los servicios de Yahoo y guardarlo en disco. El resultado de esta operación es la dirección *url* del servicio web que devuelve físicamente la imagen del mapa en cuestión.

- *doGeocode:*

Función que a partir de una dirección es capaz de calcular las coordenadas de la misma. Es un paso necesario para poder recuperar el mapa en sí en caso de que el servidor de puntos de interés no facilite las coordinadas de los puntos de interés en los que el usuario esté interesado.

5.3.4. Component_planner.php.

Tal y como explicamos anteriormente el módulo planificador del proyecto SAMAP aún no está implementado, no obstante hemos emulado su comportamiento y le hemos provisto de una interfaz para poder consumir sus recursos. Al igual que el módulo de mapas hemos creado un envoltorio a modo de componente para poder integrarlo más sencillamente con el CMS. El componente es bastante sencillo y simplemente presenta las siguientes funciones:

- *choice2xml:*

Función que convierte la colección de puntos de interés seleccionados por el usuario para formar parte del itinerario en un XML estandarizado, de tal modo que el servidor de planificación entienda la petición.

- *doPlanning:*

Función que consume el servicio web proporcionado por el servidor de planificaciones turísticas (donde la planificación se lleva a cabo consumiendo a su vez el módulo de transportes).

5.3.5. Component_googlefunctions.php.

Dado que el módulo de mapas sólo ofrecía la posibilidad de descargar mapas con el marcador de una posición, me pareció interesante la posibilidad de enriquecer la aplicación con un mapa en el cuál apareciesen todos los puntos de interés pertenecientes al itinerario y marcados según el orden indicado por el módulo planificador. Ciertamente si el usuario selecciona una gran multitud de puntos de interés para su itinerario el mapa resultará excesivamente sobrecargado, pero de un modo lógico podemos pensar que un usuario normal no seleccionará más de ocho o nueve puntos de interés para un mismo itinerario.

Dado que el servidor de mapas emplea Yahoo como proveedor geográfico para la captura de sus mapas y demás información geográfica, incluimos esta funcionalidad para mostrar la interacción con otro proveedor geográfico como es Google. A diferencia del servidor de mapas, el componente *google_functions* carece de caché o sistema de almacenamiento alguno, los mapas se crean bajo demanda cuando sea necesario mostrarlos en la aplicación web final.

El componente presenta una única función:

- *getMap:*

Función que captura un mapa del servicio de mapas de Google de acuerdo a un *array* de configuración en el que se incluyen una serie de puntos geográficos expresados en coordenadas.

5.4 Las páginas.

A la hora de articular la interacción entre el usuario y la aplicación, necesitamos una herramienta para poder implementar la lógica de la navegación en función del comportamiento del usuario. Para hacer esto, la gran parte de la lógica del negocio de la aplicación implementada recaerá en lo que hemos denominado las páginas de la aplicación. Las páginas son unidades lógicas en las que la aplicación consume aquellos recursos necesarios provenientes de los componentes y los presenta al usuario a fin de que este consuma esta información y eventualmente facilite información al sistema. Así mismo las páginas recogerán el comportamiento del usuario y lo almacenaran para su posterior uso. A continuación presentaremos las páginas que presenta la aplicación y describiremos brevemente su funcionalidad.

- *default.php*: Página de inicio de la aplicación en la cual se introducen los criterios de búsqueda y se eligen los tipos de puntos de interés que interesan al usuario. Cada vez que el usuario accede a esta página se inicializa las variables de sesión que almacenan los resultados de la búsqueda y el itinerario creado a partir de los mismos.

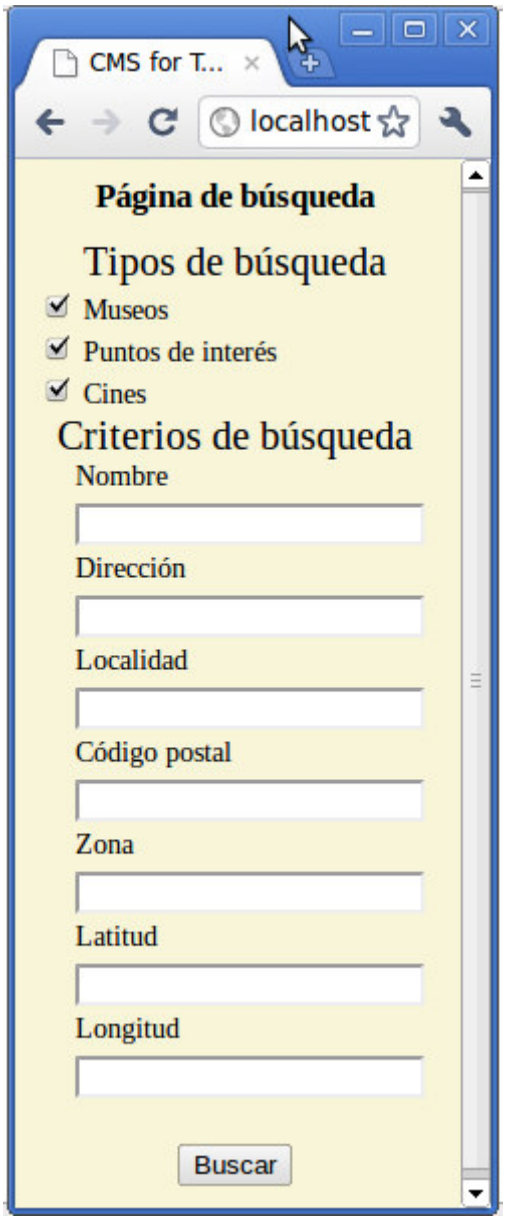

Figura 17 – Captura de la página default.

• *results.php:* Página en la cual se ejecuta la búsqueda y se muestran los resultados de la búsqueda de las atracciones turísticas existentes conforme a los criterios de búsqueda seleccionados en la página principal. Dado que el número de resultados puede ser ciertamente elevado hemos implementando un sistema de paginación de tal modo que por cada página sólo se muestren diez resultados. La lógica de la paginación se gestionará en esta página, pero será necesario contar con el apoyo del cliente para poder capturar la selección del usuario a través de las diversas páginas de resultados. Esta parte de lógica en el cliente será explicada en detalle en la sección de JavaScript. Una vez presentados dichos resultados, el usuario podrá determinar aquellos que sean realmente de su interés para posteriormente generar el itinerario.

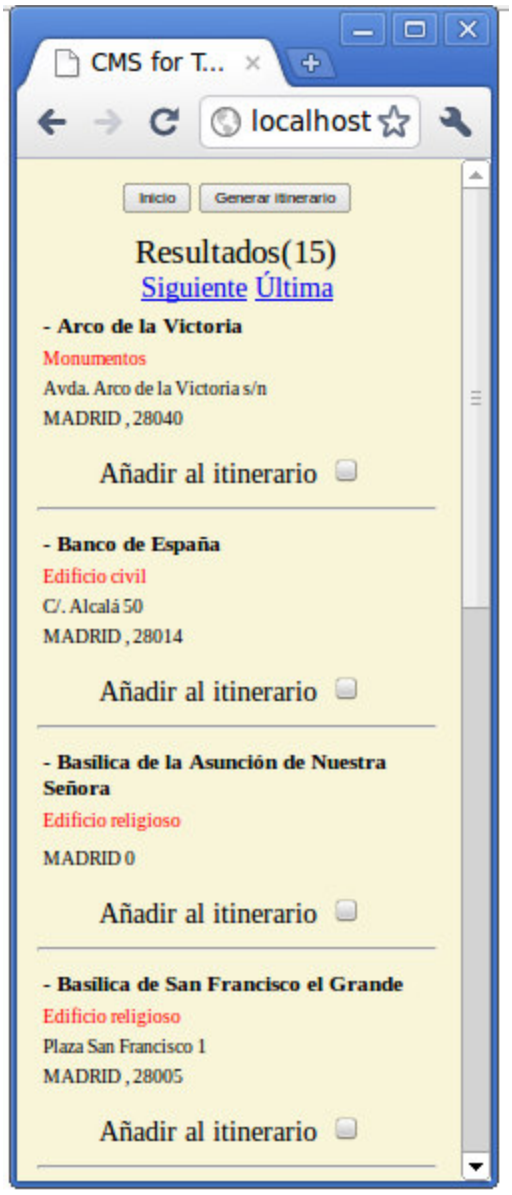

Figura 18 – Captura de la página "results.php"

• *generate.php:* Una vez seleccionados los puntos de interés que deberán aparecer en el itinerario, se mostrarán al usuario a modo de previsualización para que estos sean confirmados. Tras esta confirmación se procede a calcular el itinerario llamando al componente que gestiona la comunicación con el servidor de planificación y una vez calculado se emplea el componente que gestiona la descarga de los mapas para poder completar el itinerario con la representación gráfica de los mismos. Dada la carencia de coordenadas existentes en los puntos de interés facilitados por el servidor de puntos de interés, se realiza también en este momento una comprobación sobre ese atributo. En caso de que alguno de los puntos de interés existentes en el itinerario carezcan de

coordenadas se intenta recuperarlas del servidor de mapas a través de la operación que permite recuperar este valor a partir de la dirección postal del mismo.

Una vez tenemos toda la información necesaria mostramos el itinerario con todos los mapas a la par que damos la posibilidad de, por un lado, ver el itinerario completo en un mapa proporcionado por Google, y por otro lado ver en un mapa las transacciones entre dos puntos de interés.

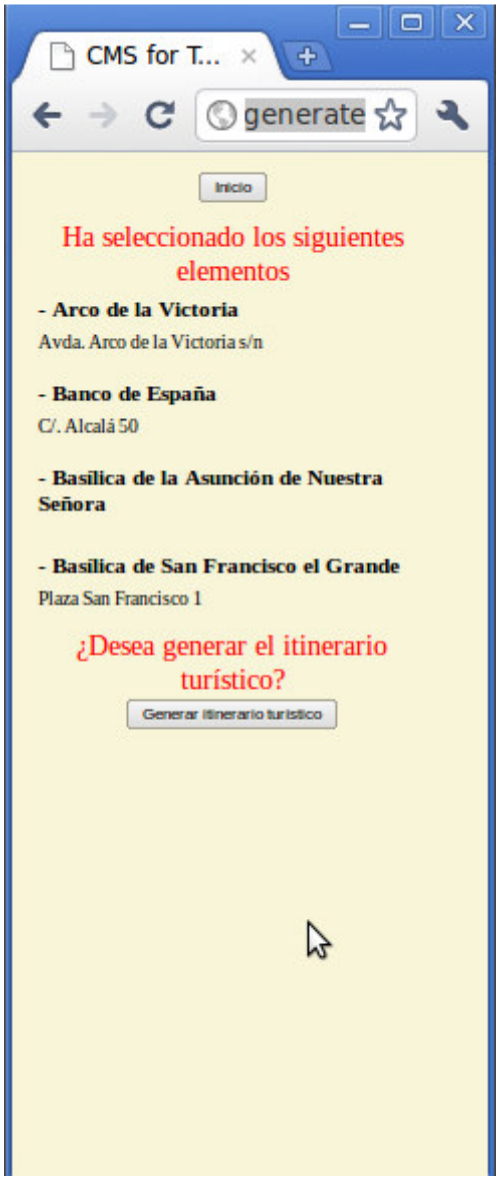

Figura 19 – Captura de la página "generate.php" I

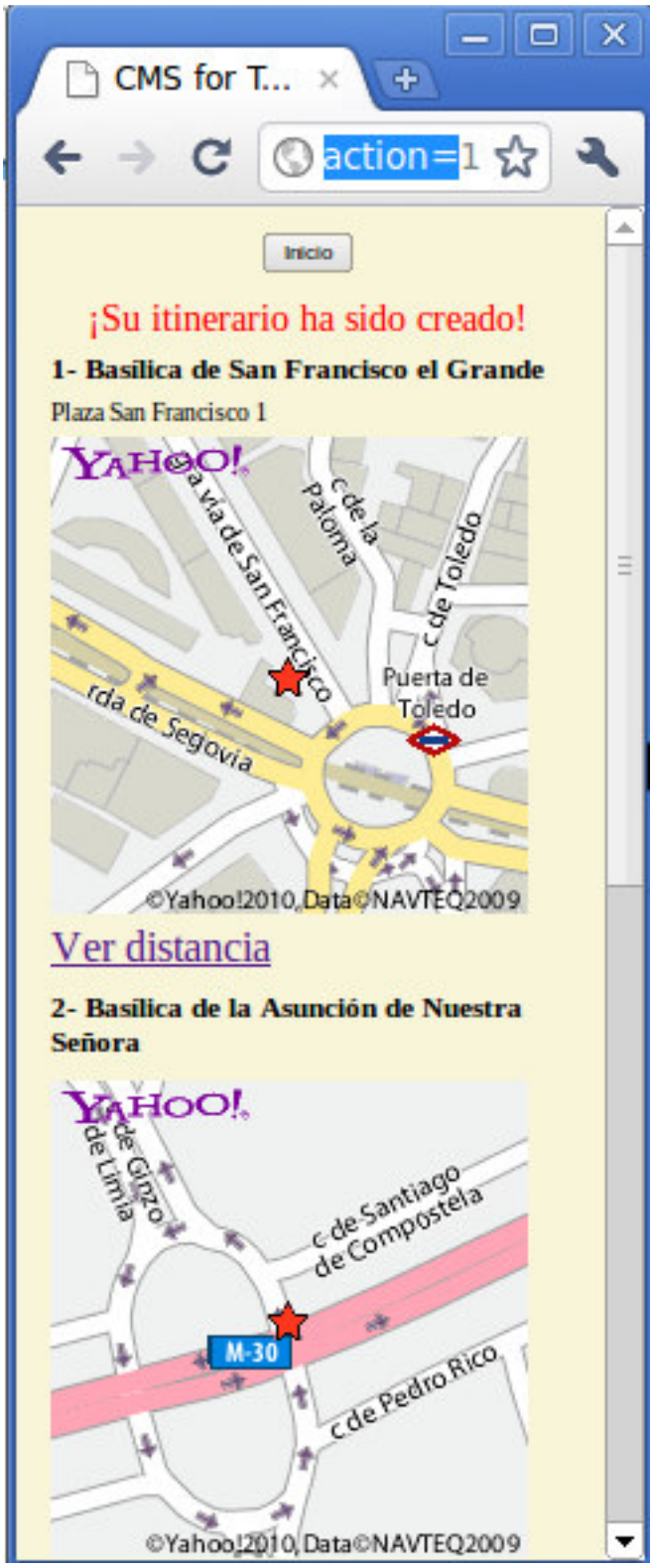

Figura 20 – Captura de la página "generate.php" II

• *google.php:* Página que simplemente emplea el componente *googlefunctions* para recuperar un mapa de Google con todos los puntos de interés pertenecientes al itinerario. Este mapa a nuestro juicio es de gran interés para el usuario dado que le

facilita una visualización global de las distancias entre los puntos de interés seleccionados y de la magnitud del itinerario.

Figura 21 – Captura de la página "google.php"

• *point2point.php:* Tal y cómo describíamos anteriormente a partir de la página *generate.php* se puede seleccionar la opción de ver las transacciones entre dos puntos del itinerario. Ciertamente para que esto sea posible los dos puntos en cuestión han de tener coordenadas, en caso contrario esta opción no estará disponible. En el caso de que ambos puntos tengan coordenadas se enlazará desde *generate.php* a esta página para mostrar la distancia entre ambos puntos.

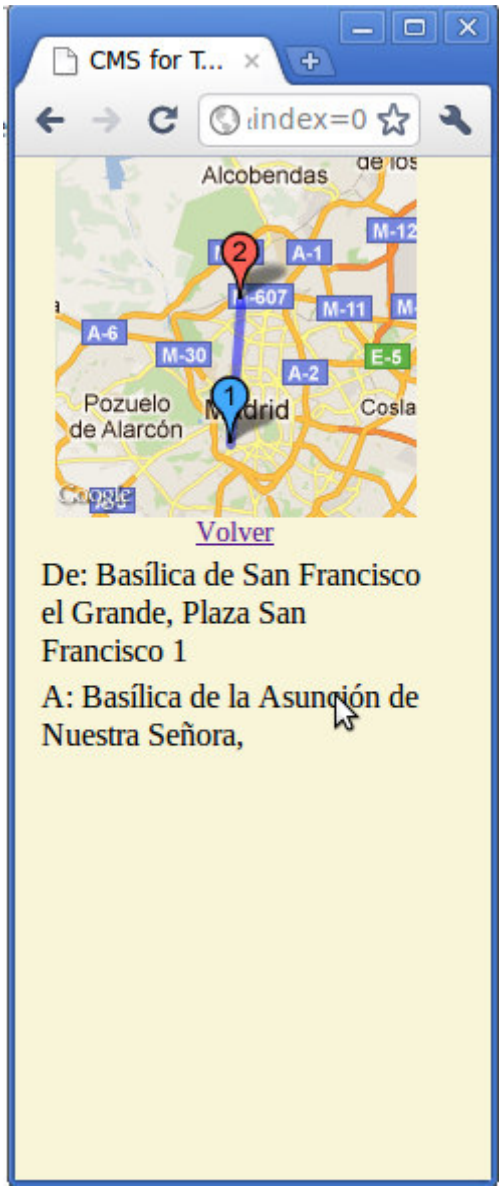

Figura 22 – Captura de la página "point2point.php"

5.5 Las plantillas.

Las plantillas son el elemento del gestor de contenidos que permite realizar el diseño de la parte gráfica de la aplicación sin necesidad de conocer la naturaleza concreta de los datos que serán mostrados por el mismo. A continuación presentaremos las plantillas empleadas por el gestor de contenidos y el contexto en el que van a ser empleados.

- all_google.tpl: La plantilla *all_google.tpl* es empleada por la página *generate.php* a fin de posibilitar el enlace con la página *google.php* donde se presentará un mapa con el itinerario completo.

- confirm.tpl: La plantilla *confirm.tpl* es empleada por la página *generate.php* para mostrar el mensaje en el que se solicita al usuario su confirmación a la hora de generar el itinerario turístico.

- main.tpl: La plantilla *main.tpl* es la plantilla principal del CMS dado que será empleada siempre por cada una de las páginas que sean navegadas. Esto es debido a que esta plantilla proporciona aquellos elementos de XHTML que son comunes a todas las páginas, y que darán un aspecto similar a todas ellas durante la navegación. Dentro de esta plantilla existirá una marca de sustitución llamada *{main}* que cada una las páginas ha de definir, para por así decirlo, dar contenido a la página. Esta plantilla crea un marco de desarrollo, dentro del cual el desarrollador gráfico podrá definir los elementos que sean necesarios.

- default.tpl: La plantilla *default.tpl* es empleada por la página *default.php* a fin de definir aquellos elementos XHTML que permitan realizar la búsqueda deseada. Básicamente se compone de un formulario donde definir los tipos de atracciones turísticas que son de interés y los criterios de búsqueda de las mismas.

- results.tpl: La plantilla *results.tpl* es empleada por la página *results.php* a fin de definir aquellos elementos XHTML que permitan visualizar los resultados de la búsqueda realizada por el usuario.

- generate.tpl: La plantilla *generate.tpl* es empleada por la página *generate.php* con un doble propósito; por un lado, presentar los elementos que el usuario ha seleccionado antes de generar propiamente el itinerario (previsualización y confirmación de los mismos) y por otro lado presentar el itinerario una vez que ha sido generado.

- google.tpl: La plantilla *google.tpl* es empleada por la página *google.php* a fin de mostrar el mapa con todos los puntos de interés del itinerario.

• map_url.tpl: La plantilla *map_url.tpl* es empleada por la página *generate.php* para definir el atributo origen de la imagen que cada punto de interés tendrá representando su mapa.

• no_results.tpl: La plantilla *no_results.tpl* es empleada por la página *results.php* para definir el mensaje de error que se le mostrará al usuario en caso de que no existan puntos de interés que coincidan con sus filtros de búsqueda dentro del sistema de almacenamiento del servidor de puntos de interés.

• point2point.tpl: La plantilla *point2point.tpl* es empleada por la página *point2point.php* para definir los elementos XHTML que mostrarán el mapa con la distancia entre dos puntos de interés dentro del itinerario.

• point2point_url.tpl: La plantilla *point2point_url.tpl* es empleada por la página *generate.php* a fin de definir el atributo de la imagen que mostrará el mapa con la distancia entre dos puntos de interés en la página *point2point.php.* El hecho de que este atributo se defina en una página que no es en la que se genera responde a la distribución lógica de la funcionalidad, ya que de este modo toda la lógica de generación de contenido recae en *generate.php,* mientras que en el resto de las página simplemente se muestra la información ya recabada y almacenada en el servidor web.

• resultlist_item.tpl: La plantilla *resultlist_item.tpl* es empleada por la página *results.php* a fin de definir los elementos de la lista que muestra el resultado de la búsqueda realizada contra el servidor de puntos de interés.

• resultlist_item_short.tpl: La plantilla *resultlist_item_short.tpl* es empleada por la página *generate.php* a fin de definir los elementos de la lista que muestra el resultado de la planificación.

5.6 Las hojas de estilo.

Las hojas de estilo permiten separar los elementos HTML de su presentación. Ciertamente se podrían definir los elementos estilísticos dentro de las plantillas pero al emplear hojas de estilo podemos por un lado realizar abstracciones, teniendo clases de estilos y por otro lado separar el contenido de la presentación.

Ciertamente en nuestra aplicación, las hojas de estilos no son excesivamente complicadas, pero lo suficientemente ilustrativas como para entender el potencial que tienen en caso de que tuviésemos una solución con mayor componente gráfico. La

implementación realizada de los estilos se basa en la definición de clases simples que son implementadas dentro del atributo *class* de los elementos XHTML.

La aplicación presenta dos hojas de estilos, una a nivel general para toda la aplicación y otra para la implementación de listas de resultados. A continuación pasamos a describirlas:

- main.css: Es la principal hoja de estilo donde se definen los estilos para los elementos comunes de la aplicación, tales como colores, anchos de pantalla, alineaciones, etc. Dado que la idea del CMS es la de tener un marco común en el que queda encuadrado el elemento *main* se definirán clases, tanto para el marco como para los elementos propios de HTML, de tal modo que gráficamente la presentación mantenga una homogeneidad a lo largo de la aplicación.
- resultlist.css: Es una hoja de estilos secundaria para definir los estilos de los elementos presentes en las listas de resultados. Es importante tener listas que al presentar los resultados sean homogéneas y puedan ser reutilizadas. Por ejemplo al mostrar los elementos seleccionados en la previsualización anterior a la generación del itinerario podemos usar los mismos estilos que al mostrar el resultado del itinerario propiamente dicho.

5.7 JavaScript.

La presencia de JavaScript en nuestra aplicación está motivada por el hecho de que cierta interacción con el usuario no necesita acudir al servidor web, sino que puede ser llevar a cabo totalmente en el lado del cliente, es decir, en el terminal.

Para el caso que nos ocupa hemos empleado JavaScript para gestionar la navegación entre las diversas páginas de resultados. La idea es simple, una vez que realizamos una búsqueda contra el servidor de puntos de interés el resultado puede ser de más de 200 puntos de interés, pero ciertamente, no queremos mostrarlos todos a la vez sino que lo iremos haciendo poco a poco, mostrando por ejemplo, diez puntos de interés. El problema reside en que el usuario irá seleccionando puntos de interés de cada una de las páginas de resultados, y el sistema ha de ser capaz de “recordar” qué puntos el usuario ha seleccionado o qué puntos que estaban seleccionados, ya no lo están.

La solución implementada se ha basado en que en el lado del cliente, cada vez que el usuario seleccionaba o deseleccionaba un punto de interés, esta información queda guardada en una variable de JavaScript y sólo cuando el usuario cambia de

página se le envía la información al servidor de qué elementos han de añadirse a la selección para crear el itinerario o qué elementos han de eliminarse de esta selección. De este modo nos ahorramos tener que comunicarnos con el servidor web cada vez que el usuario seleccione o deseleccione uno de los puntos de interés.

Todas las funciones implementadas en JavaScript están recogidas en el archivo "functions.js". A continuación presentamos las más representativas:

- updateMySession: Función que se ejecuta al seleccionar o deseleccionar uno de los puntos de interés turísticos. La idea es simplemente tener un buffer con aquellos puntos que han sido modificados (eliminados o añadidos) de tal modo que en la próxima comunicación con el servidor web se envíe esta información.
- saveSession: Función que recorre el buffer con aquellos elementos que han sido modificados y construye una dirección web con los parámetros necesarios para que estos cambios queden reflejados en el lado del servidor.

5.8 Biblioteca de funciones auxiliares.

Tal y como describimos anteriormente la mayoría de la funcionalidad en el lado del servidor está recogida en componentes que son instanciados por las páginas en función de la lógica que se necesite implementar. No obstante existe una serie de funciones auxiliares que propiamente no pertenecen a ninguno de los componentes pero que son necesarios para articular la funcionalidad del CMS. Estas funciones se encuentran agrupadas a modo de librería de funciones en un único archivo llamado "functions.php". A continuación pasamos a describir las funciones más importantes de dicha biblioteca:

- replaceArray: Función que toma como argumentos un *array* de valores y una plantilla para sustituir todas las marcas de reemplazamiento que coincidan con claves del *array*. Es especialmente útil para reemplazar estructuras repetitivas como pueden ser listas de resultados.
- createListFromArray: Función que se basa en la anterior para a partir de un *array* de resultados, crear una lista iterando sobre la función *replaceArray*.
- createPaging: Función que crea el HTML necesario para gestionar la paginación de los resultados de la búsqueda. Presenta un cierto nivel de lógica en tanto que existen diversos escenarios en lo que se refiere al número de páginas resultantes.

- simpleXml2Array: Función que convierte un XML en un *array de* PHP. El motivo de la existencia de esta función es que PHP trabaja mejor con *arrays* que con objetos XML.
- get_web_page: Función que recupera un recurso de una determinada ubicación en la red. En el contexto de nuestra aplicación ha sido empleada para consumir los servicios web externos que nos permiten comunicarnos con el resto de módulos del proyecto SAMAP.

5.9 Implementación del sistema.

5.9.1. Entorno de desarrollo.

Antes de describir cómo se ha implementado la aplicación, me parece interesante comentar el entorno de desarrollo empleado para realizar el presente proyecto de fin de carrera, ya que dada la naturaleza del mismo, el entorno de desarrollo ha ido cambiando a lo largo del mismo.

Una vez realizado el estudio del arte y el diseño a alto nivel se comenzó a analizar cada uno de los módulos que debían integrarse a través del módulo gestor de comunicaciones, pero lamentablemente al inicio de este proceso, tan sólo uno de estos módulos estaban finalizados, era el módulo servidor de puntos de interés. Al ser este un módulo desarrollado en Java y al estar yo personalmente más familiarizado con un entorno de desarrollo de tipo Windows, decidí en consenso con mi tutor desarrollar la aplicación en un entorno Windows, en concreto en mi ordenador portátil, un Acer Travelmate con Windows XP SP3 como sistema operativo, empleando PHPDesigner como entorno de desarrollo para PHP e IIS como servidor web. Una vez comencé a hacer pruebas para integrar el módulo servidor de puntos de interés turístico observé que no se comportaba como esperábamos. Por ello tuve que realizar algunas modificaciones en el código, para lo cual empleé un entorno de desarrollo para Java llamado Eclipse. El resultado de este proceso fue una primera versión del CMS, conjuntamente con la versión definitiva del componente que gestiona las comunicaciones con el módulo servidor de puntos de interés.

El problema apareció cuando a continuación pasé a estudiar el módulo de transportes. Este módulo en un principio no era demasiado relevante para el presente proyecto de fin de carrera, dado que no se comunica directamente con el módulo gestor de comunicaciones sino que la información facilitada por el mismo es consumida por el

módulo planificador. No obstante y a la vista de que el módulo planificador jamás llegó a implementarse por parte de otro compañero, nos vimos en la necesidad de emularlo, pero quisimos hacerlo de la forma más "auténtica" posible, lo cual implica que quisimos permitir la presencia del módulo de transportes en el sistema.

Por otra parte, a estas alturas del proyecto, quedó también claro que el Departamento de Informática no existía una posibilidad cómoda de encontrar un servidor donde colgar nuestras aplicaciones, y por tanto se tomó la decisión de realizar toda la implementación del proyecto en la máquina de desarrollo local.

Ante esta situación y teniendo en cuenta que el módulo de transportes se implementó para poder trabajar sólo en entornos Linux, decidimos "mudar" el desarrollo a dicha plataforma. Dado que la única máquina de la que disponía para llevar a cabo el proyecto de fin de carrera era mi ordenador portátil, tomé la decisión de trabajar en un entorno de desarrollo Linux virtual, esto es, teniendo como base la arquitectura de mi portátil, junto con su sistema operativo Windows XP, emplear una herramienta llamada VMWare para poder albergar una distribución Linux de modo virtual. VMWare es un programa que permite "reproducir" máquinas virtuales, esto quiere decir que como si de una película se tratase, a través de la herramienta VMWare Player, podemos seleccionar un fichero, el cual alberga una copia virtual de un sistema operativo completo y reproducirlo. Desde ese momento todo el desarrollo se produjo sobre una máquina virtual de VMWare con un sistema operativo Xubuntu. Dado que nos habíamos mudado a Linux, abandoné PHPDesigner y empecé a desarrollar PHP con el entorno de desarrollo NetBeans, el cuál es también válido para Java.

5.9.2. Implementación del CMS.

A continuación pasamos a describir la estructura propiamente de los archivos implementados para construir la aplicación. Partiremos de una raíz o *root* sobre la cual el resto de archivos y directorios irán montados. La idea es simple, tener en el *root* aquellos archivos que sean especiales bien por su importancia o peculiaridad y organizar el resto de un modo lógico en subdirectorios.

Por tanto, en el *root* encontraremos;

- index.php: Es el archivo principal del sistema. Se le puede considerar la puerta de la aplicación. Será el index.php quién se encargue de incluir todos los componentes del sistema y lanzar el CMS para la creación de la página en cuestión.

- settings.ini: Archivo de configuración del sistema. Ahí podremos encontrar las referencias a por ejemplo, sobre qué puerto vamos a lanzar los sockets contra el servidor de puntos de interés o cuál es el host de la aplicación.
- functions.php: Biblioteca de funciones de tipo genérico que serán usadas indistintamente a lo largo de la aplicación.
- components: Directorio donde se alojarán todos los componentes del CMS.
- javascript: Directorio donde se alojarán todos los archivos que contengan el javaScript de la aplicación.
- pages: Directorio donde se encuentran todas las páginas de la aplicación.
- styles: Directorio donde se encuentran las hojas de estilos
- templates: Directorio donde se encuentras las plantillas de la aplicación.

Manteniendo esta estructura la aplicación podría ser portada a cualquier entorno tanto Windows como Linux. Ciertamente hay que seguir una serie de pasos que describiremos a continuación en el manual de usuario.

5.10 Documentación del sistema.

5.10.1. Manual de usuario.

El presente manual de usuario tiene como fin posibilitar el uso y administración de la aplicación desarrollada para facilitar la generación y el visionado de los itinerarios turísticos que el proyecto SAMAP facilita a sus usuarios.

El administrador del sistema ha de saber que la presente aplicación interactúa con una serie de módulos en los que se reside gran parte de la lógica de la aplicación, por tanto, las tareas de administración no se reducen únicamente a la administración de la aplicación web que los usuarios consumen con sus dispositivos móviles, sino también comprenden la administración de los citados módulos y sus interfaces.

A continuación pasamos a presentar los pasos necesarios para la instalación del presente proyecto fin de carrera, así de cómo el resto de módulos con los que se comunica. El entorno de ejecución puede ser tanto Windows como Linux, no obstante, al haber sido desarrollado en una máquina con sistema operativo Ubuntu, tomaremos como ejemplo este entorno para la explicación de los pasos a seguir.

A la hora de instalar nuevos paquetes en entornos operativos de tipo GNU Linux es necesario por un lado tener permisos de administración en la máquina donde

procederemos a la instalación, así como una conexión a Internet para poder proceder a la descarga de los mismos. Existen múltiples aplicaciones que facilitan la descarga de paquetes, aunque en nuestro caso, procederemos a indicar cómo se descargan estos paquetes usando el comando "apt-get" a través de línea de comando.

5.10.1.1. Instalación de PHP 5.3

Para la instalación de PHP se debe ejecutar el comando "apt-get install php5", no obstante si desea poder ejecutar scripts de PHP desde línea de comando, podremos ejecutar también "apt-get install php5-cli". Una vez llevado a cabo este paso PHP quedará instalado en nuestra máquina. Para poder ver la configuración inicial de PHP, podemos ejecutar el comando "php –i" desde línea de comando. Dicha configuración inicial debería ser suficiente para el propósito de nuestra instalación, no obstante si considerásemos necesario llevar a cabo algún ajuste, podemos modificar el archivo de configuración de PHP que se encuentra en el directorio "/etc/php5/cli/php.ini".

Otro paquete que debemos instalar es phpcurl, el cual nos permitirá la descarga de recursos remotos. Para instalar este paquete debemos ejecutar el comando "apt-get install php5-curl".

5.10.1.2. Instalación del servidor web Apache

Para la instalación del servidor web Apache se debe ejecutar el comando "apt-get install apache2", no obstante si se va a instalar apache como módulo de php, será necesario ejecutar también "apt-get install libapache2-mod-php5". Una vez llevados a cabo estos pasos podemos reiniciar el servidor web con el comando "/etc/init.d/apache2 restart". Nada más realizar la instalación, el root de nuestro servidor web estará situado en el directorio "/var/www". El cuál podemos probar tecleando en la barra de nuestro explorador web la dirección "localhost" que debería devolvernos una página HTML de prueba.

5.10.1.3. Instalación del servidor de mapas

A la hora de instalar el servidor de mapas necesitamos copiar el directorio "filter-local" y todos sus subdirectorios dentro del directorio "mnt" de nuestra distribución Linux. El directorio "filter-local" se encuentra en el CD que acompaña el presente documento.

Dado que deseamos usar este servidor como un servicio web, necesitamos enlazar nuestro servidor web a los archivos de entrada de los servicios web que el

servidor de mapas presenta. Para ello, podemos realizar enlaces blandos desde nuestro localhost a fin de que nuestra aplicación pueda consumir dichos servicios.

Por ejemplo, para poder realizar búsquedas de geolocalización o descarga de mapas, podemos crear un enlace llamado "webservice" de la siguiente forma.

ln -s /mnt/filer-local/html/desarrollo/httpd/editores/MGM/webservice/

/var/www/webservice

De tal modo que obtengamos el siguiente enlance blando.

webservice->/mnt/filer-local/html/desarrollo/httpd/editores/MGM/webservice/

Esto implica que cada vez que queramos acceder a un servicio web del servidor de mapas, tenemos que acceder a:

"localhost/webservice/nombre_del_servicio?parametros_del_servicio"

Por ejemplo:

"http://localhost/webservice/mapas.html?longitud=-2.66860&latitud=40.84942"

Si ejecutamos este servicio, nos damos cuenta que el resultado obtenido suele tener la siguiente forma:

<?xml version="1.0" encoding="UTF-8"?>
<respuesta>
<error>0</error>
<url_mapa>**http://localhost/maps/27/1_40.84942_-2.66860_500x620.gif**</url_mapa>
</respuesta>

Esto implica que necesitamos un nuevo enlace para poder consumir los mapas, dado que el directorio "maps" no existe en nuestro sistema. Para crear este enlace ejecutamos:

ln -s /mnt/filer-local/html/desarrollo/httpd/MGM/mapas/yahoo_maps/

/var/www/maps

De este modo quedará también enlazado a nuestro servidor web los mapas que el módulo de mapas va recuperando de Yahoo.

5.10.1.4. Instalación del servidor de puntos de interés

El servidor de puntos de interés es un programa implementado en Java, por lo cual para poder recuperar los puntos de interés lo único que necesitamos es ejecutar

dicho programa, teniendo en cuenta, que en el mismo directorio donde los archivos del programa se ubiquen, hemos de tener el fichero con la ontología de los puntos de interés. Este fichero ha de llamarse "elementos.txt".

Una vez ejecutemos el programa, este se quedara a la escucha en el puerto TCP / IP correspondiente esperando la llegada de nuevas peticiones.

Los procesos internos del servidor necesitan escribir archivos temporales en el sistema de ficheros, por tanto, es necesario que el directorio tenga permisos de escritura o que en su defecto el servidor sea lanzado por un usuario con permisos suficientes.

El modo de ejecutar el programa java es:

java -jar POI.jar

5.10.1.5. Instalación del módulo planificador

Tal y como comentamos con anterioridad, el modo planificador no ha sido aún desarrollado y por tanto ha sido emulado en nuestra aplicación a fin de poder ofrecer una imagen real de lo que es todo el conjunto del proyecto SAMAP.

La emulación de la planificación se realiza en un servicio web que consta de un único archivo php llamado "planner.php". Para poder consumir este servicio debemos ubicarlo en el root de nuestro servidor web y al mismo nivel crear una carpeta llamada "files", donde las peticiones que vayan llegando se irán guardando.

5.10.1.6. Instalación del CMS

Finalmente instalaremos la presente aplicación, para ello necesitamos copiar el directorio "pfc" en el root del servidor web. Dentro de dicho directorio existe un archivo de configuración llamado *settings.ini* que ha de ser configurado, especialmente a fin de definir cuál es el host de la aplicación, dado que al ser una aplicación web con diversas páginas, el servidor web ha de conocer a que host dirigirse. Así mismo han de darse los pertinentes permisos a los archivos que componen la aplicación de tal modo que el servidor web pueda leer todos los recursos sin problema.

5.10.1.7. Uso de la aplicación

En cuanto al uso de la aplicación propiamente dicha, a día de hoy el uso es bastante sencillo. El usuario se conecta a través del terminal a la dirección de acceso de la aplicación e introduce unos criterios de búsqueda para encontrar puntos de interés turístico que sean de su interés. Una vez encontrado un conjunto de puntos que satisfagan las necesidades del usuario, esté podrá seleccionar aquellos que considere oportunos a fin de generar un itinerario turístico con datos de los puntos de interés,

mapas y rutas entre los puntos. Tal vez el único detalle a destacar sea que cada vez que el usuario accede a la página de inicio de la aplicación se eliminan los criterios de búsqueda o los itinerarios turísticos en caso de que los hubiese.

Para acceder a la aplicación hemos de definir en todo momento en qué página nos encontramos. Por ejemplo, si la aplicación se aloja en el localhost dentro del directorio "pfc", la dirección de acceso a la aplicación será:

http://localhost/pfc/index.php?page=default

5.10.2. Documentación del código fuente.

Para la documentación del código fuente se ha empleado la herramienta PhpDocumentor. Esta herramienta permite a través de una sintaxis específica insertar comentarios dentro del código fuente, los cuales son interpretados a posterioridad para crear la documentación del mismo. Dicha documentación se genera como una estructura de documentos HTML dentro de un determinado directorio, al cual se puede acceder a través del archivo "index.html" que también genera el programa.

La documentación del código fuente generada por PhpDocumentor para el presente proyecto, está accesible en el CD que acompaña al presente documento.

6. RESULTADOS

Una vez realizados el análisis, diseño e implementación del sistema hemos obtenido como resultado los siguientes elementos:

- Un sistema gestor de contenido para poder construir una aplicación web que pueda ser consumida por cualquier dispositivo móvil que tenga un navegador capaz de interpretar páginas XHTML y que tenga JavaScript.
- Una aplicación web que permite a un usuario seleccionar elementos de interés turístico dentro de la comunidad de Madrid en función de las preferencias del mismo y a partir de esa selección generar un itinerario turístico con mapas e información sobre dichos puntos de interés.
- Una colección de componentes que pueden ser empleados por el sistema gestor de contenidos para implementar las comunicaciones con el resto de módulos del proyecto SAMAP. El conjunto de estos componentes es lo que abstractamente definimos como el módulo gestor de comunicaciones del proyecto SAMAP.
- Una documentación que permite a administradores y usuarios emplear e implementar el presente proyecto, a la par que a otros desarrolladores implementar nuevas funciones y expandir la funcionalidad del mismo.
- Un sistema que puede ser implantado independientemente en un entorno Windows o en un entorno Unix. Siendo las únicas características necesarias el tener un servidor web y los suficientes permisos como para poder implantarlo.

7. CONCLUSIONES

Tras la realización de los apartados anteriores podemos llegar a las siguientes conclusiones:

- Hemos satisfecho los objetivos marcados en el presente proyecto de fin de carrera dado que hemos solucionado la parte del proyecto SAMAP que nos compete.
- Hemos demostrado que es posible integrar diversos módulos provenientes de diversos proyectos, los cuales, al ser implementados por diversos compañeros, presentaban características muy diversas a todos los niveles. Hemos conseguido realizar una abstracción para que la aplicación final vea cada uno de estos módulos como componentes de un sistema de una forma abstracta, sin tener que entrar en detalles sobre cómo están diseñados o con qué lenguaje están programados.
- Hemos profundizado en el aprendizaje de diversas tecnologías, especialmente en el campo del diseño de aplicaciones web y de la estandarización de servicios web.
- Hemos aprendido del trabajo de nuestros compañeros y nos hemos acercado a temas anteriormente desconocidos como por ejemplo la minería de datos o la implementación de sistemas de cache.
- Hemos concluido con la sensación de haber desarrollado un buen proyecto de fin de carrera que no obstante deja la puerta abierta a futuras líneas de trabajo e investigación

8. FUTURAS LÍNEAS DE TRABAJO

A la hora de analizar las futuras líneas de trabajo, debemos de hablar de cuál ha sido el principal hándicap que nos hemos encontrado a la hora de la elaboración del presente proyecto fin de carrera.

La realidad es que el conjunto del proyecto SAMAP se basa en dos pilares esenciales. Por un lado el proceso de minería de datos que nutre de información al servidor de puntos de interés turístico y otro por otro lado el módulo planificador que genera los itinerarios turísticos de acuerdo a los monumentos, cines o puntos de interés seleccionados por el usuario. Este último módulo no ha sido aún desarrollado y por este motivo una de las futuras líneas de trabajo es acometer el desarrollo e implementación del mismo, de tal modo que todas las piezas del puzle que compone el proyecto SAMAP estén en su lugar.

En lo que se refiere al proceso de minería de datos, hay que decir que los datos facilitados por el servidor de puntos de interés eran bastante defectuosos en cuanto a que estaban bastante incompletos. Por poner un ejemplo, muchas de los puntos de interés facilitados carecían de coordenadas. Esto en un principio no debería convertirse en un problema, dado que el módulo de mapas permite conocer las coordenadas de un punto a través de su dirección, pero la realidad era que la dirección también era defectuosa, sobre todo por la gran presencia de acrónimos, tales como Avda. en lugar de Avenida, C.C. en lugar de Centro Comercial, Ctra. en lugar de carretera, etc. Este hecho ha derivado en que bastantes puntos no puedan localizarse geográficamente y por tanto sus mapas no se puedan mostrar. Considero que debería reforzarse este proceso y que el módulo encargado de recuperar las atracciones turísticas de Internet, debería controlar este particular más en detalle.

Capítulo aparte merece la interfaz de búsqueda de puntos de interés facilitada por el servidor de puntos de interés, la cual se ha convertido en el cuello de botella del sistema. El problema estriba en que las búsquedas se basan en coincidencias textuales exactas, es decir, si tenemos un punto de interés ubicado en el Paseo de la Castellana 57, entresuelo e introducimos como criterio de búsqueda en el campo “dirección” el criterio “Castellana”, el sistema no encontrará ninguna coincidencia dado que para poder encontrarla deberíamos haber indicado “Paseo de la Castellana 57, entresuelo”. Este problema reduce mucho la flexibilidad de la búsqueda en el sistema. Por ejemplo, el servidor de puntos de interés permite buscar de acuerdo a la posición del punto de

interés, pero claro, nadie va a indicar como criterio de búsqueda, latitud 40.13345, longitud -3.454545.

Una posible solución que creemos convendría que fuese estudiada es el hecho de implementar un sistema de almacenamiento local para los puntos de interés. Es decir, tener por ejemplo una base de datos que a través de una tarea programada se fuese llenando de información proveniente del servidor de puntos de interés, pero que permitiese a la aplicación una interfaz de búsqueda más flexible, sobre todo basándose en conceptos como la coincidencia de parte de un campo, o en el caso de la posición, facilitando operaciones para calcular resultados dentro del radio de un punto. De este modo podríamos aprovechar el hecho de que la mayoría de los nuevos dispositivos móviles pueden facilitar la posición del mismo a través del atributo *geolocation* del navegador, para por ejemplo facilitar una lista al usuario con los puntos de interés más cercanos a su posición.

Ciertamente HTML5 es una de las líneas de trabajo futuro en lo que se refiere al desarrollo de aplicaciones web. En el caso que nos ocupa podríamos por ejemplo aprovechar el almacenamiento local del navegador, para que si por ejemplo el usuario que se encuentra de turismo en Madrid carece de conexión a Internet, darle la opción de haber realizado la generación de itinerarios turísticos con anterioridad y guardarlos en la memoria del dispositivo, de tal modo que la navegación se realice en modo *off-line* cuando se encuentre en el destino turístico.

En este sentido otra de las líneas de trabajo de futuro puede ser el hecho de implementar un módulo de gestión de usuarios en la aplicación, de tal modo que por ejemplo, el usuario al acceder al servicio se identifique de algún modo y pueda guardar sus itinerarios favoritos, marcar sus atracciones preferidas o pueda compartir sus opinión sobre los destinos visitados a través de redes sociales, etc.

En lo que se refiere al módulo de mapas, hoy en día se basa en mapas de tipo estático, es decir, imágenes que recuperamos de un proveedor como por ejemplo Yahoo, pero en un futuro, tal vez deberíamos plantearnos la posibilidad de implementar mapas dinámicos, esto es, mapas en los cuales el usuario a través de JavaScript pueda ir navegando por el mapa, haciendo zoom etc. Esta posibilidad se ha descartado a día de hoy dado que no todos los dispositivos del mercado ofrecen un hardware suficientemente potente como para que la navegación por el mapa del usuario sea satisfactoria.

9. REFERENCIAS

A lo largo del documento hemos hecho referencia a enlaces web que pueden encontrarse en las siguientes direcciones.

1. http://scalab.uc3m.es/~dborrajo/samap/
2. http://www.mofuse.com/
3. http://www.infogin.com/
4. http://www.momac.net/
5. http://www.baturamobile.com/
6. http://www.traveldodo.com/
7. http://www.hsl.fi/EN/Pages/default.aspx
8. http://www.gsmworld.com/newsroom/resources/g.htm
9. http://www.gsmworld.com/
10. http://www.3gpp.org/About-3GPP
11. http://www.w3.org/MarkUp/#xhtml11
12. http://www.librosweb.es/javascript/capitulo1/breve_historia.html

9.1 Fuentes consultadas.

Para consultas sobre el lenguaje de programación Java se ha consultado la siguiente fuente:

- Aaron E. Walsh, John Fronckowiak *Java Bible* John Wiley & Sons, Inc., 1998

Para consultas sobre el lenguaje de programación PHP se han consultado las siguientes fuentes:

- Minera, Francisco José *Proyectos con PHP* MP Ediciones 2005
- Ramos Monso, Martín *Programación PHP* MP Ediciones 2004

10. ILUSTRACIONES

11. TABLAS